Program Analysis (70020)

Data Flow Analysis

Herbert Wiklicky

Department of Computing
Imperial College London

herbert@doc.ic.ac.uk
h.wiklicky@imperial.ac.uk

Autumn 2024

Data Flow Analysis

The general approach for determining program properties for procedural languages via a dataflow analysis:

Data Flow Analysis

The general approach for determining program properties for procedural languages via a dataflow analysis:

- Extract (Control) Flow Information

Data Flow Analysis

The general approach for determining program properties for procedural languages via a dataflow analysis:

- Extract (Control) Flow Information
- Formulate Data Flow Equations

Data Flow Analysis

The general approach for determining program properties for procedural languages via a dataflow analysis:

- Extract (Control) Flow Information
- Formulate Data Flow Equations

- Construct Solution(s) of Equations

Data Flow Analysis

The general approach for determining program properties for procedural languages via a dataflow analysis:

- Extract (Control) Flow Information
- Formulate Data Flow Equations
  - Update Local Information
- Construct Solution(s) of Equations

Data Flow Analysis

The general approach for determining program properties for procedural languages via a dataflow analysis:

- Extract (Control) Flow Information
- Formulate Data Flow Equations
  - Update Local Information
  - Collect Global Information
- Construct Solution(s) of Equations

Available Expressions

The *Available Expressions Analysis* will determine:

> *For each program point, which expressions must (are guaranteed to) have already been computed, and not later modified, on all paths to that program point.*

Available Expressions

The *Available Expressions Analysis* will determine:

> *For each program point, which expressions must (are guaranteed to) have already been computed, and not later modified, on all paths to that program point.*

This information can be used to avoid the re-computation of an expression. For clarity, we will concentrate on arithmetic expressions.

Example

Consider the following simple program:

$$
\begin{array}{l}
[\ x := a + b\ ]^1; \\
[\ y := a * b\ ]^2; \\
\textbf{while}\ [y > a + b]^3\ \textbf{do}\ (\\
\qquad [\ a := a + 1\ ]^4; \\
\qquad [\ x := a + b\ ]^5\ )
\end{array}
$$

Example

Consider the following simple program:

$$
\begin{array}{l}
[\ x := a + b\ ]^1; \\
[\ y := a * b\ ]^2; \\
\textbf{while}\ [y > a + b]^3\ \textbf{do}\ (\\
\qquad [\ a := a + 1\ ]^4; \\
\qquad [\ x := a + b\ ]^5\ )
\end{array}
$$

It should be clear that the expression a+b is available every time the execution reaches the test (label 3) in the loop; as a consequence, the expression need not be recomputed.

AE Analysis

$$\textit{kill}_{\mathsf{AE}} : \textbf{Block}_\star \rightarrow \mathcal{P}(\textbf{AExp}_\star)$$

AE Analysis

$$\mathit{kill}_{\mathsf{AE}} : \mathbf{Block}_{\star} \to \mathcal{P}(\mathbf{AExp}_{\star})$$

$$\mathit{gen}_{\mathsf{AE}} : \mathbf{Block}_{\star} \to \mathcal{P}(\mathbf{AExp}_{\star})$$

AE Analysis

$$kill_{\mathrm{AE}} : \mathbf{Block}_\star \rightarrow \mathcal{P}(\mathbf{AExp}_\star)$$

$$gen_{\mathrm{AE}} : \mathbf{Block}_\star \rightarrow \mathcal{P}(\mathbf{AExp}_\star)$$

$$\mathrm{AE}_{entry} : \mathbf{Lab}_\star \rightarrow \mathcal{P}(\mathbf{AExp}_\star)$$

AE Analysis

$$kill_{\mathsf{AE}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

$$gen_{\mathsf{AE}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

$$\mathsf{AE}_{entry} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

$$\mathsf{AE}_{exit} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

AE Auxiliary Functions

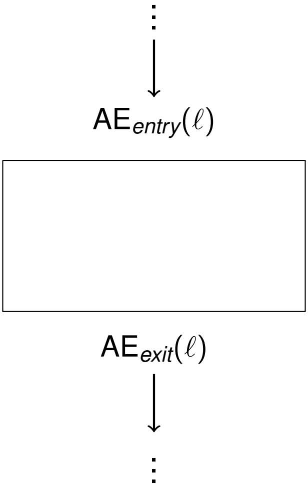
$$\begin{array}{rcl} kill_{\mathsf{AE}}([\, x := a \,]^{\ell}) & = & \{a' \in \mathbf{AExp}_{\star} \mid x \in FV(a')\} \\ kill_{\mathsf{AE}}([\, \mathbf{skip} \,]^{\ell}) & = & \emptyset \\ kill_{\mathsf{AE}}([b]^{\ell}) & = & \emptyset \end{array}$$

AE Auxiliary Functions

$$
\begin{array}{rcl}
kill_{\mathsf{AE}}([\, x := a \,]^{\ell}) & = & \{a' \in \mathbf{AExp}_{\star} \mid x \in FV(a')\} \\
kill_{\mathsf{AE}}([\, \mathbf{skip} \,]^{\ell}) & = & \emptyset \\
kill_{\mathsf{AE}}([b]^{\ell}) & = & \emptyset
\end{array}
$$

$$
\begin{array}{rcl}
gen_{\mathsf{AE}}([\, x := a \,]^{\ell}) & = & \{a' \in \mathbf{AExp}(a) \mid x \notin FV(a')\} \\
gen_{\mathsf{AE}}([\, \mathbf{skip} \,]^{\ell}) & = & \emptyset \\
gen_{\mathsf{AE}}([b]^{\ell}) & = & \mathbf{AExp}(b)
\end{array}
$$

AE Local Change (e.g. expression $x + y$)



AE Local Change (e.g. expression $x + y$)

$\vdots$

$\downarrow$

$\text{AE}_{entry}(\ell)$

$[\ z := x + y\ ]^{\ell}\ \surd$

$\text{AE}_{exit}(\ell)$

$\downarrow$

$\vdots$

AE Local Change (e.g. expression $x + y$)

$$\vdots$$

$$\downarrow$$

$$\mathrm{AE}_{entry}(\ell)$$

$$[x + y < x]^{\ell} \quad \surd$$

$$\mathrm{AE}_{exit}(\ell)$$

$$\downarrow$$

$$\vdots$$

AE Local Change (e.g. expression $x + y$)

$$\vdots$$

$$\downarrow$$

$$\mathrm{AE}_{entry}(\ell)$$

$$[\; {\color{red} x} := x + y \;]^{\ell} \quad {\color{red}\times}$$

$$\mathrm{AE}_{exit}(\ell)$$

$$\downarrow$$

$$\vdots$$

AE Local Change (e.g. expression $x + y$)

$$\vdots$$

$$\downarrow$$

$$\text{AE}_{entry}(\ell)$$

$$[x + y < x]^{\ell} \quad \surd$$
$$[\ z := x + y\ ]^{\ell} \quad \surd$$
$$[\ x := x + y\ ]^{\ell} \quad \times$$

$$\text{AE}_{exit}(\ell)$$

$$\downarrow$$

$$\vdots$$

Whenever a variable x in an expression gets a new value the expression becomes unavailable.

AE Equation Schemes

$$\mathsf{AE}_{\mathit{entry}}(\ell) \;=\; \begin{cases} \emptyset, \text{ if } \ell = \mathit{init}(S_\star) \\ \bigcap\{\mathsf{AE}_{\mathit{exit}}(\ell') \mid (\ell', \ell) \in \mathit{flow}(S_\star)\}, \text{ otherwise} \end{cases}$$

AE Equation Schemes

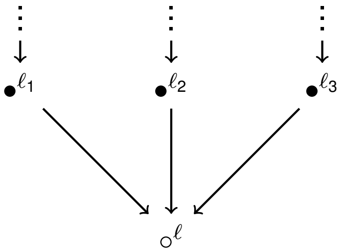
$$\mathsf{AE}_{entry}(\ell) \;=\; \begin{cases} \emptyset, \text{ if } \ell = init(S_\star) \\ \bigcap\{\mathsf{AE}_{exit}(\ell') \mid (\ell', \ell) \in flow(S_\star)\}, \text{ otherwise} \end{cases}$$

$$\mathsf{AE}_{exit}(\ell) \;=\; (\mathsf{AE}_{entry}(\ell) \backslash kill_{\mathsf{AE}}([B]^\ell)) \cup gen_{\mathsf{AE}}(([B]^\ell) \text{ where } [B]^\ell \in blocks(S_\star)$$

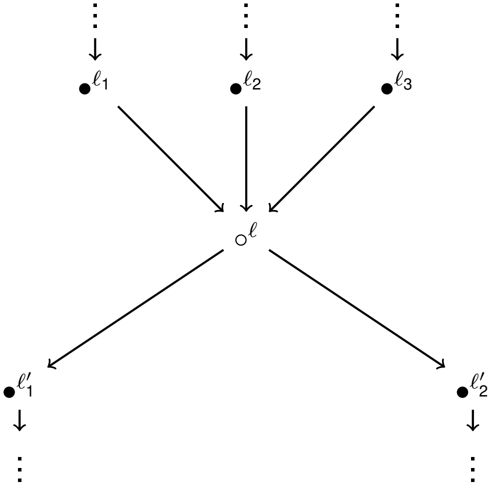
AE Global Collection

$\circ^{\ell}$

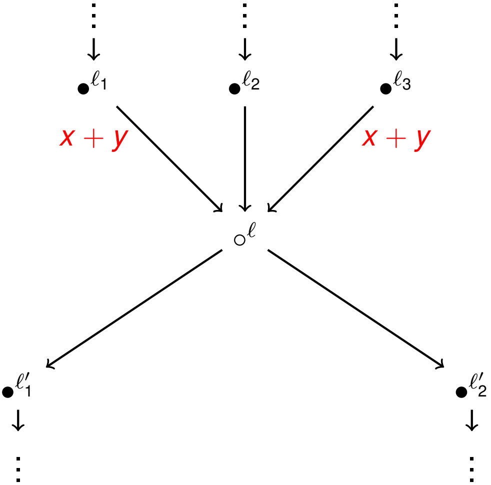
AE Global Collection



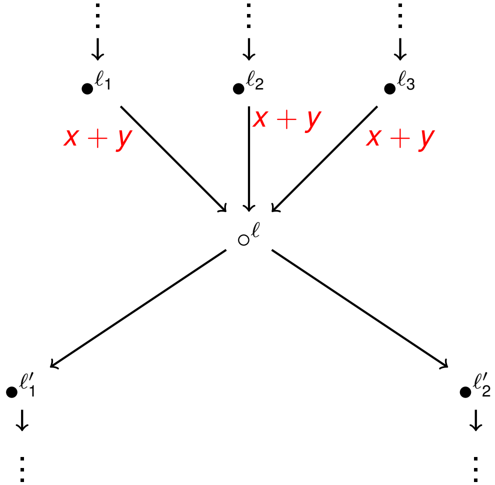
AE Global Collection



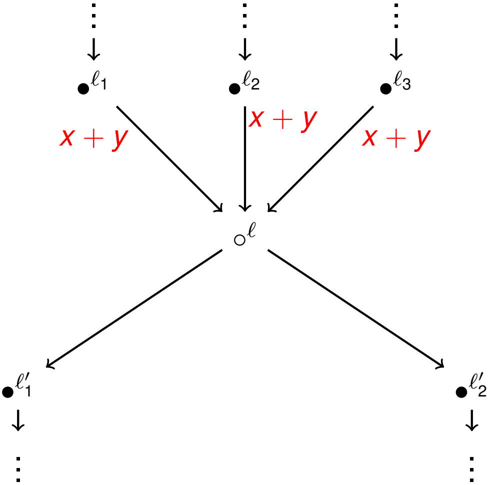
AE Global Collection



AE Global Collection



AE Global Collection



We push information “forward in time”.

Largest Solution

The analysis is a *forward analysis* and we are interested in the *largest* sets satisfying the equation for AE_{entry} and AE_{exit}.

Largest Solution

The analysis is a *forward analysis* and we are interested in the *largest* sets satisfying the equation for AE_{entry} and AE_{exit}.

$$[\ z := x + y\ ]^{\ell};\textbf{while}\ [true]^{\ell'}\ \textbf{do}\ [\ \textbf{skip}\ ]^{\ell''}$$

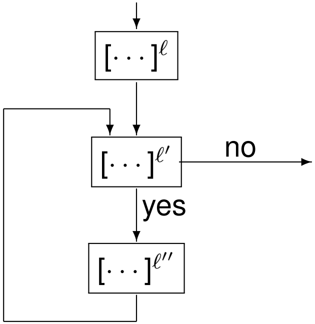
Largest Solution

The analysis is a *forward analysis* and we are interested in the *largest* sets satisfying the equation for AE_{entry} and AE_{exit}.

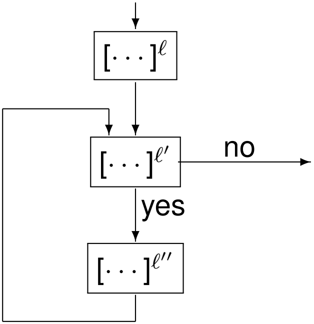
$$[\ z := x + y\ ]^{\ell};\textbf{while}\ [true]^{\ell'}\ \textbf{do}\ [\ \textbf{skip}\ ]^{\ell''}$$

$$\begin{array}{rcl}
\mathsf{AE}_{entry}(\ell) & = & \emptyset \\
\mathsf{AE}_{entry}(\ell') & = & \mathsf{AE}_{exit}(\ell)\ \cap\ \mathsf{AE}_{exit}(\ell'') \\
\mathsf{AE}_{entry}(\ell'') & = & \mathsf{AE}_{exit}(\ell') \\
\mathsf{AE}_{exit}(\ell) & = & \mathsf{AE}_{entry}(\ell) \cup \{x + y\} \\
\mathsf{AE}_{exit}(\ell') & = & \mathsf{AE}_{entry}(\ell') \\
\mathsf{AE}_{exit}(\ell'') & = & \mathsf{AE}_{entry}(\ell'')
\end{array}$$

Obtaining Solutions



Obtaining Solutions



After some simplification, we find that:

$$\mathsf{AE}_{entry}(\ell') = \{x + y\} \cap \mathsf{AE}_{entry}(\ell')$$

AE Example

$$
\begin{array}{l}
[\ x := a + b\ ]^1; \\
[\ y := a * b\ ]^2; \\
\textbf{while}\ [y > a + b]^3\ \textbf{do}\ (\\
\qquad [\ a := a + 1\ ]^4; \\
\qquad [\ x := a + b\ ]^5\ )
\end{array}
$$

AE Example

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

ℓ	$kill_{\mathsf{AE}}(\ell)$	$gen_{\mathsf{AE}}(\ell)$
1	$\emptyset$	$\{a + b\}$
2	$\emptyset$	$\{a * b\}$
3	$\emptyset$	$\{a + b\}$
4	$\{a + b, a * b, a + 1\}$	$\emptyset$
5	$\emptyset$	$\{a + b\}$

AE Example: Equations

$$
\begin{array}{l}
[\ x := a + b\ ]^1; \\
[\ y := a * b\ ]^2; \\
\textbf{while}\ [y > a + b]^3\ \textbf{do}\ (\\
\qquad [\ a := a + 1\ ]^4; \\
\qquad [\ x := a + b\ ]^5\ )
\end{array}
$$

AE Example: Equations

$$
\begin{array}{l}
[\ x := a + b\ ]^1;\\
[\ y := a * b\ ]^2;\\
\textbf{while}\ [y > a + b]^3\ \textbf{do}\ (\\
\qquad [\ a := a + 1\ ]^4;\\
\qquad [\ x := a + b\ ]^5\ )
\end{array}
$$

$$
\begin{aligned}
\mathrm{AE}_{exit}(1) &= \mathrm{AE}_{entry}(1) \cup \{a + b\}\\
\mathrm{AE}_{exit}(2) &= \mathrm{AE}_{entry}(2) \cup \{a * b\}\\
\mathrm{AE}_{exit}(3) &= \mathrm{AE}_{entry}(3) \cup \{a + b\}\\
\mathrm{AE}_{exit}(4) &= \mathrm{AE}_{entry}(4) \backslash \{a + b, a * b, a + 1\}\\
\mathrm{AE}_{exit}(5) &= \mathrm{AE}_{entry}(5) \cup \{a + b\}
\end{aligned}
$$

AE Example: Equations

$$
\begin{array}{l}
[\ x := a + b\ ]^1; \\
[\ y := a * b\ ]^2; \\
\textbf{while}\ [y > a + b]^3\ \textbf{do}\ (\\
\qquad [\ a := a + 1\ ]^4; \\
\qquad [\ x := a + b\ ]^5\ )
\end{array}
$$

AE Example: Equations

$$
\begin{array}{l}
[\ x := a + b\ ]^1; \\
[\ y := a * b\ ]^2; \\
\textbf{while}\ [y > a + b]^3\ \textbf{do}\ (\\
\qquad [\ a := a + 1\ ]^4; \\
\qquad [\ x := a + b\ ]^5\ )
\end{array}
$$

$$
\begin{array}{rcl}
\mathrm{AE}_{entry}(1) & = & \emptyset \\
\mathrm{AE}_{entry}(2) & = & \mathrm{AE}_{exit}(1) \\
\mathrm{AE}_{entry}(3) & = & \mathrm{AE}_{exit}(2) \cap \mathrm{AE}_{exit}(5) \\
\mathrm{AE}_{entry}(4) & = & \mathrm{AE}_{exit}(3) \\
\mathrm{AE}_{entry}(5) & = & \mathrm{AE}_{exit}(4)
\end{array}
$$

AE Example: Equations

$$
\begin{aligned}
\mathrm{AE}_{entry}(1) &= \emptyset \\
\mathrm{AE}_{entry}(2) &= \mathrm{AE}_{exit}(1) \\
\mathrm{AE}_{entry}(3) &= \mathrm{AE}_{exit}(2) \cap \mathrm{AE}_{exit}(5) \\
\mathrm{AE}_{entry}(4) &= \mathrm{AE}_{exit}(3) \\
\mathrm{AE}_{entry}(5) &= \mathrm{AE}_{exit}(4)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{AE}_{exit}(1) &= \mathrm{AE}_{entry}(1) \cup \{a+b\} \\
\mathrm{AE}_{exit}(2) &= \mathrm{AE}_{entry}(2) \cup \{a*b\} \\
\mathrm{AE}_{exit}(3) &= \mathrm{AE}_{entry}(3) \cup \{a+b\} \\
\mathrm{AE}_{exit}(4) &= \mathrm{AE}_{entry}(4) \backslash \{a+b, a*b, a+1\} \\
\mathrm{AE}_{exit}(5) &= \mathrm{AE}_{entry}(5) \cup \{a+b\}
\end{aligned}
$$

AE Example: Solutions

ℓ	$\mathsf{AE}_{entry}(\ell)$	$\mathsf{AE}_{exit}(\ell)$
1	$\emptyset$	$\{a + b\}$
2	$\{a + b\}$	$\{a + b, a * b\}$
3	$\{a + b\}$	$\{a + b\}$
4	$\{a + b\}$	$\emptyset$
5	$\emptyset$	$\{a + b\}$

AE Example: Solutions

ℓ	$\text{AE}_{entry}(\ell)$	$\text{AE}_{exit}(\ell)$
1	$\emptyset$	$\{a+b\}$
2	$\{a+b\}$	$\{a+b, a*b\}$
3	$\{a+b\}$	$\{a+b\}$
4	$\{a+b\}$	$\emptyset$
5	$\emptyset$	$\{a+b\}$

$[\ x := a+b\ ]^1;$
$[\ y := a*b\ ]^2;$
while $[y > a+b]^3$ **do** (
$\quad [\ a := a+1\ ]^4;$
$\quad [\ x := a+b\ ]^5$)

AE Example: Solutions

ℓ	$\mathsf{AE}_{entry}(\ell)$	$\mathsf{AE}_{exit}(\ell)$
1	$\emptyset$	$\{a+b\}$
2	$\{a+b\}$	$\{a+b, a*b\}$
3	$\{a+b\}$	$\{a+b\}$
4	$\{a+b\}$	$\emptyset$
5	$\emptyset$	$\{a+b\}$

Note that, even though a is redefined in the loop, the expression a+b is re-evaluated in the loop and so it is always available on entry to the loop. On the other hand, a*b is available on the first entry to the loop but is killed before the next iteration.

Reaching Definitions Analysis

The *Reaching Definitions Analysis* is analogous to the previous one except that we are interested in:

> *For each program point, which assignments* may *have been made and not overwritten, when program execution reaches this point along some path.*

Reaching Definitions Analysis

The *Reaching Definitions Analysis* is analogous to the previous one except that we are interested in:

> *For each program point, which assignments may have been made and not overwritten, when program execution reaches this point along some path.*

A main application of Reaching Definitions Analysis is in the construction of direct links between blocks that produce values and blocks that use them.

Example

A simple example to illustrate the *RD* analysis would be:

$$
\begin{array}{l}
[\ x := 5\ ]^1; \\
[\ y := 1\ ]^2; \\
\textbf{while}\ [x > 1]^3\ \textbf{do}\ (\\
\qquad [\ y := x * y\ ]^4; \\
\qquad [\ x := x - 1\ ]^5\ )
\end{array}
$$

Example

A simple example to illustrate the *RD* analysis would be:

$$
\begin{array}{l}
[\ x := 5\ ]^1; \\
[\ y := 1\ ]^2; \\
\textbf{while}\ [x > 1]^3\ \textbf{do}\ (\\
\qquad [\ y := x * y\ ]^4; \\
\qquad [\ x := x - 1\ ]^5\ )
\end{array}
$$

All of the assignments reach the entry of 4 (the assignments labelled 1 and 2 reach there on the first iteration); only the assignments labelled 1, 4 and 5 reach the entry of 5.

RD Analysis

$$\mathit{kill}_{\mathsf{RD}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$$

RD Analysis

$$kill_{\mathrm{RD}} : \mathbf{Block}_\star \rightarrow \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$$

$$gen_{\mathrm{RD}} : \mathbf{Block}_\star \rightarrow \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$$

RD Analysis

$$kill_{\mathrm{RD}} : \mathbf{Block}_{\star} \rightarrow \mathcal{P}(\mathbf{Var}_{\star} \times \mathbf{Lab}_{\star})$$

$$gen_{\mathrm{RD}} : \mathbf{Block}_{\star} \rightarrow \mathcal{P}(\mathbf{Var}_{\star} \times \mathbf{Lab}_{\star})$$

$$\mathrm{RD}_{entry} : \mathbf{Lab}_{\star} \rightarrow \mathcal{P}(\mathbf{Var}_{\star} \times \mathbf{Lab}_{\star})$$

RD Analysis

$$kill_{\mathrm{RD}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$$

$$gen_{\mathrm{RD}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$$

$$\mathrm{RD}_{entry} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$$

$$\mathrm{RD}_{exit} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$$

RD Analysis

$$kill_{\mathrm{RD}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$$

$$gen_{\mathrm{RD}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$$

$$\mathrm{RD}_{entry} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$$

$$\mathrm{RD}_{exit} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star)$$

Remark: Strictly speaking we need $\mathcal{P}(\mathbf{Var}_\star \times (\mathbf{Lab}_\star \cup \{?\}))$.

RD Auxiliary Functions

$$
\begin{array}{rcl}
\mathit{kill}_{\mathrm{RD}}([\ x := a\ ]^{\ell}) & = & \{(x,?)\} \cup \{(x,\ell') \mid \\
 & & [B]^{\ell'} \text{ a "definition" of } x \text{ in } S_\star\} \\
\mathit{kill}_{\mathrm{RD}}([\ \textbf{skip}\ ]^{\ell}) & = & \emptyset \\
\mathit{kill}_{\mathrm{RD}}([b]^{\ell}) & = & \emptyset
\end{array}
$$

RD Auxiliary Functions

$$
\begin{aligned}
\mathit{kill}_{\mathrm{RD}}([\, x := a \,]^{\ell}) &= \{(x, ?)\} \cup \{(x, \ell') \mid \\
&\qquad [B]^{\ell'} \text{ a “definition” of } x \text{ in } S_\star\} \\
\mathit{kill}_{\mathrm{RD}}([\, \textbf{skip} \,]^{\ell}) &= \emptyset \\
\mathit{kill}_{\mathrm{RD}}([b]^{\ell}) &= \emptyset
\end{aligned}
$$

$$
\begin{aligned}
\mathit{gen}_{\mathrm{RD}}([\, x := a \,]^{\ell}) &= \{(x, \ell)\} \\
\mathit{gen}_{\mathrm{RD}}([\, \textbf{skip} \,]^{\ell}) &= \emptyset \\
\mathit{gen}_{\mathrm{RD}}([b]^{\ell}) &= \emptyset
\end{aligned}
$$

RD Equation Schemes

$$\mathrm{RD}_{entry}(\ell) = \begin{cases} \{(x, ?) \mid x \in FV(S_\star)\}, & \text{if } \ell = init(S_\star) \\ \bigcup\{\mathrm{RD}_{exit}(\ell') \mid (\ell', \ell) \in flow(S_\star)\}, & \text{otherwise} \end{cases}$$

RD Equation Schemes

$$\mathsf{RD}_{entry}(\ell) = \begin{cases} \{(x, ?) \mid x \in FV(S_\star)\}, & \text{if } \ell = init(S_\star) \\ \bigcup\{\mathsf{RD}_{exit}(\ell') \mid (\ell', \ell) \in flow(S_\star)\}, & \text{otherwise} \end{cases}$$

$$\begin{aligned} \mathsf{RD}_{exit}(\ell) &= (\mathsf{RD}_{entry}(\ell) \backslash kill_{\mathsf{RD}}([B]^\ell)) \cup gen_{\mathsf{RD}}([B]^\ell) \\ &\quad \text{where } [B]^\ell \in blocks(S_\star) \end{aligned}$$

Smallest Solution

Similar to before, this is a *forward analysis* but we are interested in the *smallest* sets satisfying the equation for RD_{entry}.

Smallest Solution

Similar to before, this is a *forward analysis* but we are interested in the *smallest* sets satisfying the equation for RD_{entry}.

$$[\ z := x + y\ ]^{\ell}; \textbf{while}\ [true]^{\ell'}\ \textbf{do}\ [\ \textbf{skip}\ ]^{\ell''}$$

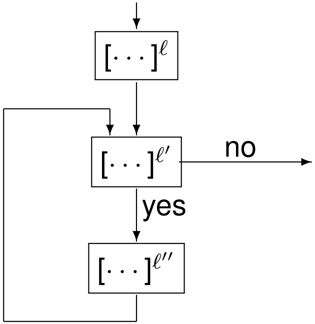
Smallest Solution

Similar to before, this is a *forward analysis* but we are interested in the *smallest* sets satisfying the equation for RD_{entry}.

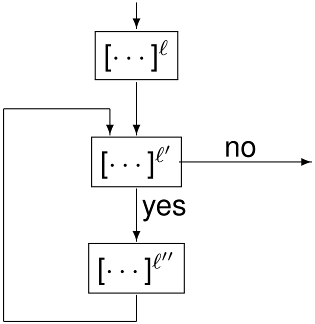
$$[\ z := x + y\ ]^{\ell};\textbf{while}\ [true]^{\ell'}\ \textbf{do}\ [\ \textbf{skip}\ ]^{\ell''}$$

$$
\begin{aligned}
\mathsf{RD}_{entry}(\ell) &= \{(x,?),(y,?),(z,?)\}\\
\mathsf{RD}_{entry}(\ell') &= \mathsf{RD}_{exit}(\ell) \cup \mathsf{RD}_{exit}(\ell'')\\
\mathsf{RD}_{entry}(\ell'') &= \mathsf{RD}_{exit}(\ell')\\
\mathsf{RD}_{exit}(\ell) &= (\mathsf{RD}_{entry}(\ell)\setminus\{(z,?)\}) \cup \{(z,\ell)\}\\
\mathsf{RD}_{exit}(\ell') &= \mathsf{RD}_{entry}(\ell')\\
\mathsf{RD}_{exit}(\ell'') &= \mathsf{RD}_{entry}(\ell'')
\end{aligned}
$$

Obtaining Solutions



Obtaining Solutions



After some simplification, we find that:

$$\mathsf{RD}_{entry}(\ell') = \{(x, ?), (y, ?), (z, \ell)\} \cup \mathsf{RD}_{entry}(\ell')$$

RD Variations

Sometimes, when the Reaching Definitions analysis is presented in the literature, one has $\mathrm{RD}_{entry}(init(S_\star)) = \emptyset$ rather than $\mathrm{RD}_{entry}(init(S_\star)) = \{((x, ?) \mid x \in FV(S_\star)\}$.

RD Variations

Sometimes, when the Reaching Definitions analysis is presented in the literature, one has $\mathrm{RD}_{entry}(init(S_\star)) = \emptyset$ rather than $\mathrm{RD}_{entry}(init(S_\star)) = \{((x, ?) \mid x \in FV(S_\star)\}$.

This is correct only for programs that always assign variables before their first use; incorrect optimisations may result if this is not the case. The advantage of our formulation is that it is always semantically sound.

RD Example

$$
\begin{array}{l}
[\ x := 5\ ]^1; \\
[\ y := 1\ ]^2; \\
\textbf{while}\ [x > 1]^3\ \textbf{do}\ (\\
\qquad [\ y := x * y\ ]^4; \\
\qquad [\ x := x - 1\ ]^5\ )
\end{array}
$$

RD Example

$[\ x := 5\ ]^1;$
$[\ y := 1\ ]^2;$
$\textbf{while}\ [x > 1]^3\ \textbf{do}\ ($
$\quad [\ y := x * y\ ]^4;$
$\quad [\ x := x - 1\ ]^5\ )$

ℓ	$kill_{\mathsf{RD}}(\ell)$	$gen_{\mathsf{RD}}(\ell)$
1	$\{(x,?),(x,1),(x,5)\}$	$\{(x,1)\}$
2	$\{(y,?),(y,2),(y,4)\}$	$\{(y,2)\}$
3	$\emptyset$	$\emptyset$
4	$\{(y,?),(y,2),(y,4)\}$	$\{(y,4)\}$
5	$\{(x,?),(x,1),(x,5)\}$	$\{(x,5)\}$

RD Example: Equations

$[\ x := 5\ ]^1;$
$[\ y := 1\ ]^2;$
while $[x > 1]^3$ **do** (
    $[\ y := x * y\ ]^4;$
    $[\ x := x - 1\ ]^5$)

RD Example: Equations

$$
\begin{array}{l}
[\ x := 5\ ]^1;\\
[\ y := 1\ ]^2;\\
\textbf{while}\ [x > 1]^3\ \textbf{do}\ (\\
\qquad [\ y := x * y\ ]^4;\\
\qquad [\ x := x - 1\ ]^5\ )
\end{array}
$$

$$
\begin{aligned}
\mathrm{RD}_{exit}(1) &= (\mathrm{RD}_{entry}(1) \backslash \{(x, ?), (x, 1), (x, 5)\}) \cup \{(x, 1)\}\\
\mathrm{RD}_{exit}(2) &= (\mathrm{RD}_{entry}(2) \backslash \{(y, ?), (y, 2), (y, 4)\}) \cup \{(y, 2)\}\\
\mathrm{RD}_{exit}(3) &= \mathrm{RD}_{entry}(3)\\
\mathrm{RD}_{exit}(4) &= (\mathrm{RD}_{entry}(4) \backslash \{(y, ?), (y, 2), (y, 4)\}) \cup \{(y, 4)\}\\
\mathrm{RD}_{exit}(5) &= (\mathrm{RD}_{entry}(5) \backslash \{(x, ?), (x, 1), (x, 5)\}) \cup \{(x, 5)\}
\end{aligned}
$$

RD Example: Equations

$[\ x := 5\ ]^1;$
$[\ y := 1\ ]^2;$
while $[x > 1]^3$ **do** (
$\quad [\ y := x * y\ ]^4;$
$\quad [\ x := x - 1\ ]^5$)

RD Example: Equations

$$
\begin{array}{l}
[\ x := 5\ ]^1; \\
[\ y := 1\ ]^2; \\
\textbf{while}\ [x > 1]^3\ \textbf{do}\ (\\
\qquad [\ y := x * y\ ]^4; \\
\qquad [\ x := x - 1\ ]^5\ )
\end{array}
$$

$$
\begin{aligned}
\mathrm{RD}_{entry}(1) &= \{(x, ?), (y, ?)\} \\
\mathrm{RD}_{entry}(2) &= \mathrm{RD}_{exit}(1) \\
\mathrm{RD}_{entry}(3) &= \mathrm{RD}_{exit}(2) \cup \mathrm{RD}_{exit}(5) \\
\mathrm{RD}_{entry}(4) &= \mathrm{RD}_{exit}(3) \\
\mathrm{RD}_{entry}(5) &= \mathrm{RD}_{exit}(4)
\end{aligned}
$$

RD Example: Equations

$$
\begin{aligned}
\mathrm{RD}_{entry}(1) &= \{(x,?),(y,?)\} \\
\mathrm{RD}_{entry}(2) &= \mathrm{RD}_{exit}(1) \\
\mathrm{RD}_{entry}(3) &= \mathrm{RD}_{exit}(2) \cup \mathrm{RD}_{exit}(5) \\
\mathrm{RD}_{entry}(4) &= \mathrm{RD}_{exit}(3) \\
\mathrm{RD}_{entry}(5) &= \mathrm{RD}_{exit}(4)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{RD}_{exit}(1) &= (\mathrm{RD}_{entry}(1) \backslash \{(x,?),(x,1),(x,5)\}) \cup \{(x,1)\} \\
\mathrm{RD}_{exit}(2) &= (\mathrm{RD}_{entry}(2) \backslash \{(y,?),(y,2),(y,4)\}) \cup \{(y,2)\} \\
\mathrm{RD}_{exit}(3) &= \mathrm{RD}_{entry}(3) \\
\mathrm{RD}_{exit}(4) &= (\mathrm{RD}_{entry}(4) \backslash \{(y,?),(y,2),(y,4)\}) \cup \{(y,4)\} \\
\mathrm{RD}_{exit}(5) &= (\mathrm{RD}_{entry}(5) \backslash \{(x,?),(x,1),(x,5)\}) \cup \{(x,5)\}
\end{aligned}
$$

RD Example: Solutions

ℓ	$\mathrm{RD}_{entry}(\ell)$	$\mathrm{RD}_{exit}(\ell)$
1	$\{(x,?),(y,?)\}$	$\{(y,?),(x,1)\}$
2	$\{(y,?),(x,1)\}$	$\{(x,1),(y,2)\}$
3	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$
4	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(y,4),(x,5)\}$
5	$\{(x,1),(y,4),(x,5)\}$	$\{(y,4),(x,5)\}$

RD Example: Solutions

ℓ	$\mathrm{RD}_{entry}(\ell)$	$\mathrm{RD}_{exit}(\ell)$
1	$\{(x,?),(y,?)\}$	$\{(y,?),(x,1)\}$
2	$\{(y,?),(x,1)\}$	$\{(x,1),(y,2)\}$
3	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$
4	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(y,4),(x,5)\}$
5	$\{(x,1),(y,4),(x,5)\}$	$\{(y,4),(x,5)\}$

$[\ x := 5\ ]^1;$
$[\ y := 1\ ]^2;$
while $[x > 1]^3$ **do** (
$[\ y := x * y\ ]^4;$
$[\ x := x - 1\ ]^5$)

Very Busy Expression Analysis

An expression is *very busy* at the exit from a label if, no matter what path is taken from the label, the expression *must* (is guaranteed to) always be used before any of the variables occurring in it are redefined. The aim of the *Very Busy Expressions Analysis* is to determine:

> *For each program point, which expressions* must *(is guaranteed to) be very busy at exit from the point.*

Very Busy Expression Analysis

An expression is *very busy* at the exit from a label if, no matter what path is taken from the label, the expression *must* (is guaranteed to) always be used before any of the variables occurring in it are redefined. The aim of the *Very Busy Expressions Analysis* is to determine:

> *For each program point, which expressions* must *(is guaranteed to) be very busy at exit from the point.*

A possible optimisation based on this information is to evaluate the expression at the block and store its value for later use; this optimisation is sometimes called *hoisting* the expression.

Example

We illustrate this analysis with the following example:

$$
\begin{array}{l}
\textbf{if } [a > b]^1 \\
\quad \textbf{then } (\ [\ x := b - a\ ]^2; \\
\quad \qquad\quad\ \ [\ y := a - b\ ]^3\ ) \\
\quad \textbf{else } (\ [\ y := b - a\ ]^4; \\
\quad \qquad\quad\ \ [\ x := a - b\ ]^5\ )
\end{array}
$$

Example

We illustrate this analysis with the following example:

$$
\begin{array}{l}
\textbf{if } [a > b]^1 \\
\quad \textbf{then } (\ [\ x := b - a\ ]^2; \\
\qquad\qquad [\ y := a - b\ ]^3\ ) \\
\quad \textbf{else } (\ [\ y := b - a\ ]^4; \\
\qquad\qquad [\ x := a - b\ ]^5\ )
\end{array}
$$

The expressions $a - b$ and $b - a$ are both very busy at the start of the program (label 1). They can be hoisted resulting in a code size reduction.

VB Analysis

$$\mathit{kill}_{\mathsf{VB}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

VB Analysis

$$kill_{\mathsf{VB}} : \mathbf{Block}_\star \rightarrow \mathcal{P}(\mathbf{AExp}_\star)$$

$$gen_{\mathsf{VB}} : \mathbf{Block}_\star \rightarrow \mathcal{P}(\mathbf{AExp}_\star)$$

VB Analysis

$$kill_{\text{VB}} : \textbf{Block}_\star \to \mathcal{P}(\textbf{AExp}_\star)$$

$$gen_{\text{VB}} : \textbf{Block}_\star \to \mathcal{P}(\textbf{AExp}_\star)$$

$$\text{VB}_{entry} : \textbf{Lab}_\star \to \mathcal{P}(\textbf{AExp}_\star)$$

VB Analysis

$$kill_{\mathrm{VB}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

$$gen_{\mathrm{VB}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

$$\mathrm{VB}_{entry} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

$$\mathrm{VB}_{exit} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

VB Analysis

$$kill_{\mathrm{VB}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

$$gen_{\mathrm{VB}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

$$\mathrm{VB}_{entry} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

$$\mathrm{VB}_{exit} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

The analysis is a *backward analysis* and we are interested in the *largest* sets satisfying the equation for VB_{exit}.

VB Auxiliary Functions

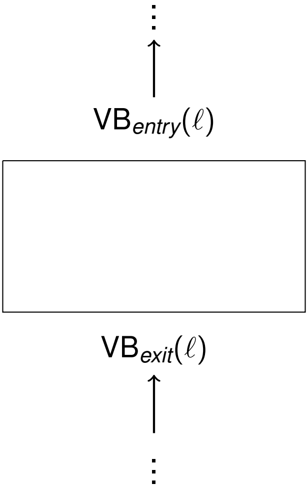
$$
\begin{aligned}
\mathit{kill}_{\mathrm{VB}}([\, x := a \,]^{\ell}) &= \{a' \in \mathbf{AExp}_{\star} \mid x \in \mathit{FV}(a')\} \\
\mathit{kill}_{\mathrm{VB}}([\, \mathbf{skip} \,]^{\ell}) &= \emptyset \\
\mathit{kill}_{\mathrm{VB}}([b]^{\ell}) &= \emptyset
\end{aligned}
$$

VB Auxiliary Functions

$$\begin{array}{rcl} kill_{\mathsf{VB}}([\, x := a\,]^{\ell}) & = & \{a' \in \mathbf{AExp}_{\star} \mid x \in FV(a')\} \\ kill_{\mathsf{VB}}([\, \mathbf{skip}\,]^{\ell}) & = & \emptyset \\ kill_{\mathsf{VB}}([b]^{\ell}) & = & \emptyset \end{array}$$

$$\begin{array}{rcl} gen_{\mathsf{VB}}([\, x := a\,]^{\ell}) & = & \mathbf{AExp}(a) \\ gen_{\mathsf{VB}}([\, \mathbf{skip}\,]^{\ell}) & = & \emptyset \\ gen_{\mathsf{VB}}([b]^{\ell}) & = & \mathbf{AExp}(b) \end{array}$$

VB Local Change



VB Local Change

$\vdots$

$\uparrow$

$\mathsf{VB}_{entry}(\ell)$

$[\ z := x + y\ ]^{\ell}\ \surd$

$\mathsf{VB}_{exit}(\ell)$

$\uparrow$

$\vdots$

VB Local Change

$$\vdots$$

$$\uparrow$$

$$\mathrm{VB}_{entry}(\ell)$$

$$[x + y < x]^{\ell} \quad \surd$$

$$\mathrm{VB}_{exit}(\ell)$$

$$\uparrow$$

$$\vdots$$

VB Local Change

$$\vdots$$

$$\uparrow$$

$$\mathrm{VB}_{entry}(\ell)$$

$$[\ x := x + y\ ]^{\ell} \ \surd$$

$$\mathrm{VB}_{exit}(\ell)$$

$$\uparrow$$

$$\vdots$$

VB Local Change

$$\vdots$$

$$\uparrow$$

$$\mathrm{VB}_{entry}(\ell)$$

$$\begin{array}{rl} [x + y < x]^\ell & \surd \\ [\ {\color{blue}z} := x + y\ ]^\ell & \surd \\ [\ {\color{red}x} := x + y\ ]^\ell & \surd \end{array}$$

$$\mathrm{VB}_{exit}(\ell)$$

$$\uparrow$$

$$\vdots$$

Whenever a variable x in an expression gets a new value it does not help us if it was evaluated before.

VB Equation Schemes

$$\mathsf{VB}_{exit}(\ell) \quad = \quad \begin{cases} \emptyset, \text{if } \ell \in \mathit{final}(S_\star) \\ \bigcap\{\mathsf{VB}_{entry}(\ell') \mid (\ell', \ell) \in \mathit{flow}^R(S_\star)\}, \text{otherwise} \end{cases}$$

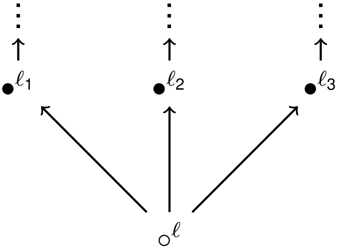
VB Equation Schemes

$$\mathsf{VB}_{exit}(\ell) = \begin{cases} \emptyset, \text{if } \ell \in \mathit{final}(S_\star) \\ \bigcap\{\mathsf{VB}_{entry}(\ell') \mid (\ell', \ell) \in \mathit{flow}^R(S_\star)\}, \text{otherwise} \end{cases}$$

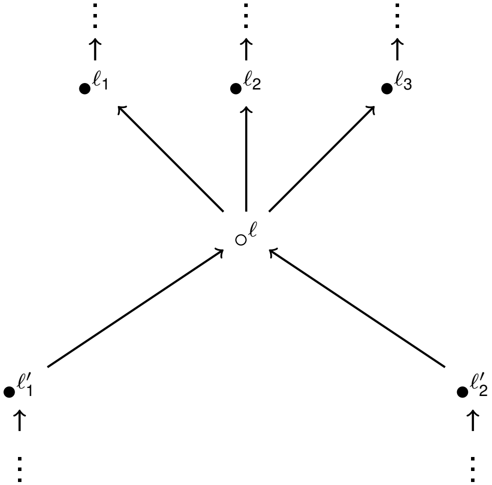
$$\mathsf{VB}_{entry}(\ell) = (\mathsf{VB}_{exit}(\ell) \backslash \mathit{kill}_{\mathsf{VB}}([B]^\ell)) \cup \mathit{gen}_{\mathsf{VB}}(B^\ell) \\ \text{where } [B]^\ell \in \mathit{blocks}(S_\star)$$

VB Global Collection

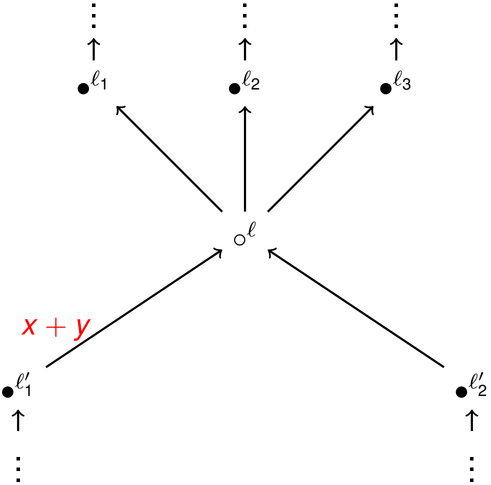
VB Global Collection



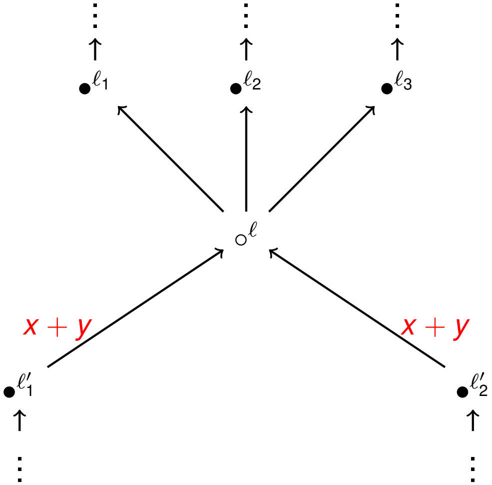
VB Global Collection



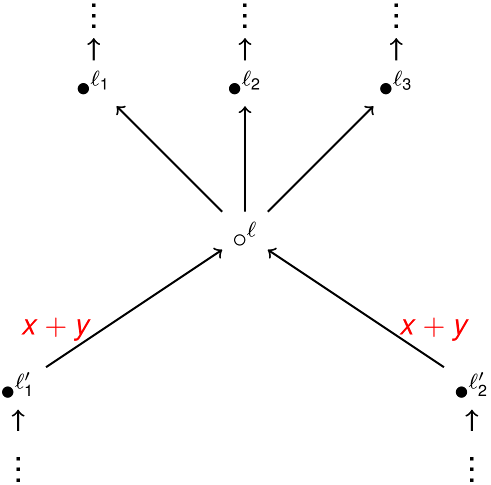
VB Global Collection



VB Global Collection



VB Global Collection



We need to go “back in time”.

VB Example

$$
\begin{array}{l}
\textbf{if } [a > b]^1 \\
\quad \textbf{then } (\ [\ x := b - a\ ]^2; \\
\qquad\qquad\ \ [\ y := a - b\ ]^3\ ) \\
\quad \textbf{else } (\ [\ y := b - a\ ]^4; \\
\qquad\qquad\ \ [\ x := a - b\ ]^5\ )
\end{array}
$$

VB Example

$$
\begin{array}{l}
\textbf{if } [a > b]^1 \\
\quad \textbf{then } (\ [\ x := b - a\ ]^2; \\
\qquad\qquad [\ y := a - b\ ]^3\ ) \\
\quad \textbf{else } (\ [\ y := b - a\ ]^4; \\
\qquad\qquad [\ x := a - b\ ]^5\ )
\end{array}
$$

ℓ	$kill_{\mathrm{VB}}(\ell)$	$gen_{\mathrm{VB}}(\ell)$
1	$\emptyset$	$\emptyset$
2	$\emptyset$	$\{b - a\}$
3	$\emptyset$	$\{a - b\}$
4	$\emptyset$	$\{b - a\}$
5	$\emptyset$	$\{a - b\}$

VB Example: Equations

$$
\begin{array}{l}
\textbf{if } [a > b]^1 \\
\quad \textbf{then } (\,[\, x := b - a \,]^2; \\
\qquad\qquad\; [\, y := a - b \,]^3 \,) \\
\quad \textbf{else } (\,[\, y := b - a \,]^4; \\
\qquad\qquad\; [\, x := a - b \,]^5 \,)
\end{array}
$$

VB Example: Equations

$$
\begin{array}{l}
\textbf{if } [a > b]^1 \\
\quad \textbf{then } (\ [\ x := b - a\ ]^2; \\
\qquad\qquad [\ y := a - b\ ]^3\ ) \\
\quad \textbf{else } (\ [\ y := b - a\ ]^4; \\
\qquad\qquad [\ x := a - b\ ]^5\ )
\end{array}
$$

$$
\begin{array}{rcl}
\mathrm{VB}_{entry}(1) & = & \mathrm{VB}_{exit}(1) \\
\mathrm{VB}_{entry}(2) & = & \mathrm{VB}_{exit}(2) \cup \{b - a\} \\
\mathrm{VB}_{entry}(3) & = & \{a - b\} \\
\mathrm{VB}_{entry}(4) & = & \mathrm{VB}_{exit}(4) \cup \{b - a\} \\
\mathrm{VB}_{entry}(5) & = & \{a - b\}
\end{array}
$$

VB Example: Equations

$$
\begin{aligned}
&\textbf{if } [a > b]^1 \\
&\quad \textbf{then } (\ [\ x := b - a\ ]^2; \\
&\qquad\qquad [\ y := a - b\ ]^3\ ) \\
&\quad \textbf{else } (\ [\ y := b - a\ ]^4; \\
&\qquad\qquad [\ x := a - b\ ]^5\ )
\end{aligned}
$$

VB Example: Equations

$$
\begin{array}{l}
\textbf{if } [a > b]^1 \\
\quad \textbf{then } (\, [\, x := b - a \,]^2; \\
\qquad\qquad [\, y := a - b \,]^3 \,) \\
\quad \textbf{else } (\, [\, y := b - a \,]^4; \\
\qquad\qquad [\, x := a - b \,]^5 \,)
\end{array}
$$

$$
\begin{array}{rcl}
\mathrm{VB}_{exit}(1) & = & \mathrm{VB}_{entry}(2) \cap \mathrm{VB}_{entry}(4) \\
\mathrm{VB}_{exit}(2) & = & \mathrm{VB}_{entry}(3) \\
\mathrm{VB}_{exit}(3) & = & \emptyset \\
\mathrm{VB}_{exit}(4) & = & \mathrm{VB}_{entry}(5) \\
\mathrm{VB}_{exit}(5) & = & \emptyset
\end{array}
$$

VB Example: Equations

$$
\begin{aligned}
\mathrm{VB}_{entry}(1) &= \mathrm{VB}_{exit}(1) \\
\mathrm{VB}_{entry}(2) &= \mathrm{VB}_{exit}(2) \cup \{b - a\} \\
\mathrm{VB}_{entry}(3) &= \{a - b\} \\
\mathrm{VB}_{entry}(4) &= \mathrm{VB}_{exit}(4) \cup \{b - a\} \\
\mathrm{VB}_{entry}(5) &= \{a - b\}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{VB}_{exit}(1) &= \mathrm{VB}_{entry}(2) \cap \mathrm{VB}_{entry}(4) \\
\mathrm{VB}_{exit}(2) &= \mathrm{VB}_{entry}(3) \\
\mathrm{VB}_{exit}(3) &= \emptyset \\
\mathrm{VB}_{exit}(4) &= \mathrm{VB}_{entry}(5) \\
\mathrm{VB}_{exit}(5) &= \emptyset
\end{aligned}
$$

VB Example: Solutions

ℓ	$\mathrm{VB}_{entry}(\ell)$	$\mathrm{VB}_{exit}(\ell)$
1	$\{a - b, b - a\}$	$\{a - b, b - a\}$
2	$\{a - b, b - a\}$	$\{a - b\}$
3	$\{a - b\}$	$\emptyset$
4	$\{a - b, b - a\}$	$\{a - b\}$
5	$\{a - b\}$	$\emptyset$

VB Example: Solutions

ℓ	$\mathrm{VB}_{entry}(\ell)$	$\mathrm{VB}_{exit}(\ell)$
1	$\{a - b, b - a\}$	$\{a - b, b - a\}$
2	$\{a - b, b - a\}$	$\{a - b\}$
3	$\{a - b\}$	$\emptyset$
4	$\{a - b, b - a\}$	$\{a - b\}$
5	$\{a - b\}$	$\emptyset$

if $[a > b]^1$
  then ($[\ x := b - a\ ]^2$;
    $[\ y := a - b\ ]^3$)
  else ($[\ y := b - a\ ]^4$;
    $[\ x := a - b\ ]^5$)

Live Variable Analysis

A variable is *live* at the exit from a label if there exists a path from the label to a use of the variable that does not re-define the variable. The *Live Variables Analysis* will determine:

> *For each program point, which variables* may *be live at the exit from the point.*

Live Variable Analysis

A variable is *live* at the exit from a label if there exists a path from the label to a use of the variable that does not re-define the variable. The *Live Variables Analysis* will determine:

> *For each program point, which variables* may *be live at the exit from the point.*

This analysis might be used as the basis for *Dead Code Elimination*. If the variable is not live at the exit from a label then, if the elementary block is an assignment to the variable, the elementary block can be eliminated.

Example

The example program to illustrate the *LV* analysis is:

$$
\begin{array}{l}
[\, x := 2 \,]^1; \\
[\, y := 4 \,]^2; \\
[\, x := 1 \,]^3; \\
(\, \textbf{if } [y > x]^4 \\
\quad \textbf{then } [\, z := y \,]^5 \\
\quad \textbf{else } [\, z := y * y \,]^6 \,); \\
[\, x := z \,]^7
\end{array}
$$

Example

The example program to illustrate the *LV* analysis is:

$$
\begin{array}{l}
[\, x := 2 \,]^1; \\
[\, y := 4 \,]^2; \\
[\, x := 1 \,]^3; \\
(\ \textbf{if}\ [y > x]^4 \\
\quad \textbf{then}\ [\, z := y \,]^5 \\
\quad \textbf{else}\ [\, z := y * y \,]^6\ ); \\
[\, x := z \,]^7
\end{array}
$$

The variable x is not live at the exit from 1; the first assignment to x is thus redundant and can be eliminated. Both x and y are alive at the exit from label 3.

LV Analysis

$$kill_{\mathsf{LV}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$

LV Analysis

$$kill_{\mathsf{LV}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$

$$gen_{\mathsf{LV}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$

LV Analysis

$$\mathit{kill}_{\mathrm{LV}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$

$$\mathit{gen}_{\mathrm{LV}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$

$$\mathrm{LV}_{\mathit{entry}} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$

LV Analysis

$$kill_{\mathsf{LV}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$

$$gen_{\mathsf{LV}} : \mathbf{Block}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$

$$\mathsf{LV}_{entry} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$

$$\mathsf{LV}_{exit} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{Var}_\star)$$

LV Analysis

$$kill_{\mathrm{LV}} : \mathbf{Block}_\star \rightarrow \mathcal{P}(\mathbf{Var}_\star)$$

$$gen_{\mathrm{LV}} : \mathbf{Block}_\star \rightarrow \mathcal{P}(\mathbf{Var}_\star)$$

$$\mathrm{LV}_{entry} : \mathbf{Lab}_\star \rightarrow \mathcal{P}(\mathbf{Var}_\star)$$

$$\mathrm{LV}_{exit} : \mathbf{Lab}_\star \rightarrow \mathcal{P}(\mathbf{Var}_\star)$$

The analysis is a *backward analysis* and we are interested in the *smallest* sets satisfying the equation for LV_{exit}.

LV Auxiliary Functions

$$
\begin{aligned}
\mathit{kill}_{\mathrm{LV}}([\, x := a \,]^{\ell}) &= \{x\} \\
\mathit{kill}_{\mathrm{LV}}([\, \textbf{skip} \,]^{\ell}) &= \emptyset \\
\mathit{kill}_{\mathrm{LV}}([b]^{\ell}) &= \emptyset
\end{aligned}
$$

LV Auxiliary Functions

$$
\begin{aligned}
\mathit{kill}_{\mathrm{LV}}([\,x := a\,]^{\ell}) &= \{x\} \\
\mathit{kill}_{\mathrm{LV}}([\,\mathbf{skip}\,]^{\ell}) &= \emptyset \\
\mathit{kill}_{\mathrm{LV}}([b]^{\ell}) &= \emptyset
\end{aligned}
$$

$$
\begin{aligned}
\mathit{gen}_{\mathrm{LV}}([\,x := a\,]^{\ell}) &= \mathit{FV}(a) \\
\mathit{gen}_{\mathrm{LV}}([\,\mathbf{skip}\,]^{\ell}) &= \emptyset \\
\mathit{gen}_{\mathrm{LV}}([b]^{\ell}) &= \mathit{FV}(b)
\end{aligned}
$$

LV Equation Schemes

$$\mathsf{LV}_{\mathit{exit}}(\ell) = \begin{cases} \emptyset, \text{if } \ell \in \mathit{final}(S_\star) \\ \bigcup\{\mathsf{LV}_{\mathit{entry}}(\ell') \mid (\ell', \ell) \in \mathit{flow}^R(S_\star)\}, \text{otherwise} \end{cases}$$

LV Equation Schemes

$$\mathsf{LV}_{exit}(\ell) = \begin{cases} \emptyset, \text{if } \ell \in \mathit{final}(S_\star) \\ \bigcup\{\mathsf{LV}_{entry}(\ell') \mid (\ell', \ell) \in \mathit{flow}^R(S_\star)\}, \text{otherwise} \end{cases}$$

$$\mathsf{LV}_{entry}(\ell) = (\mathsf{LV}_{exit}(\ell) \backslash \mathit{kill}_{\mathsf{LV}}([B]^\ell) \cup \mathit{gen}_{\mathsf{LV}}([B]^\ell)$$
$$\text{where } [B]^\ell \in \mathit{blocks}(S_\star)$$

LV Example

$$[\ x := 2\ ]^1;\ [\ y := 4\ ]^2;\ [\ x := 1\ ]^3;$$
$$(\textbf{if}\ \ [y > x]^4\ \textbf{then}\ \ [\ z := y\ ]^5\ \textbf{else}\ \ [\ z := y * y\ ]^6\ );$$
$$[\ x := z\ ]^7$$

LV Example

$[\ x := 2\ ]^1;\ [\ y := 4\ ]^2;\ [\ x := 1\ ]^3;$
$(\textbf{if}\ \ [y > x]^4\ \textbf{then}\ \ [\ z := y\ ]^5\ \textbf{else}\ \ [\ z := y * y\ ]^6\ );$
$[\ x := z\ ]^7$

ℓ	$kill_{\mathrm{LV}}(\ell)$	$gen_{\mathrm{LV}}(\ell)$
1	$\{x\}$	$\emptyset$
2	$\{y\}$	$\emptyset$
3	$\{x\}$	$\emptyset$
4	$\emptyset$	$\{x, y\}$
5	$\{z\}$	$\{y\}$
6	$\{z\}$	$\{y\}$
7	$\{x\}$	$\{z\}$

LV Example: Equations

$$[\ x := 2\ ]^1;\ [\ y := 4\ ]^2;\ [\ x := 1\ ]^3;$$
$$(\textbf{if}\ [y > x]^4\ \textbf{then}\ [\ z := y\ ]^5\ \textbf{else}\ [\ z := y * y\ ]^6\ );$$
$$[\ x := z\ ]^7$$

LV Example: Equations

$[\ x := 2\ ]^1;\ [\ y := 4\ ]^2;\ [\ x := 1\ ]^3;$
$(\textbf{if}\ [y > x]^4\ \textbf{then}\ [\ z := y\ ]^5\ \textbf{else}\ [\ z := y * y\ ]^6\ );$
$[\ x := z\ ]^7$

$$
\begin{aligned}
LV_{entry}(1) &= LV_{exit}(1) \setminus \{x\} \\
LV_{entry}(2) &= LV_{exit}(2) \setminus \{y\} \\
LV_{entry}(3) &= LV_{exit}(3) \setminus \{x\} \\
LV_{entry}(4) &= LV_{exit}(4) \cup \{x, y\} \\
LV_{entry}(5) &= (LV_{exit}(5) \setminus \{z\}) \cup \{y\} \\
LV_{entry}(6) &= (LV_{exit}(6) \setminus \{z\}) \cup \{y\} \\
LV_{entry}(7) &= \{z\}
\end{aligned}
$$

LV Example: Equations

$[\ x := 2\ ]^1;\ [\ y := 4\ ]^2;\ [\ x := 1\ ]^3;$
$(\textbf{if}\ \ [y > x]^4\ \textbf{then}\ \ [\ z := y\ ]^5\ \textbf{else}\ \ [\ z := y * y\ ]^6\ );$
$[\ x := z\ ]^7$

LV Example: Equations

$[\ x := 2\ ]^1;\ [\ y := 4\ ]^2;\ [\ x := 1\ ]^3;$
$(\textbf{if}\ \ [y > x]^4\ \textbf{then}\ \ [\ z := y\ ]^5\ \textbf{else}\ \ [\ z := y * y\ ]^6\ );$
$[\ x := z\ ]^7$

$$
\begin{aligned}
\mathrm{LV}_{exit}(1) &= \mathrm{LV}_{entry}(2)\\
\mathrm{LV}_{exit}(2) &= \mathrm{LV}_{entry}(3)\\
\mathrm{LV}_{exit}(3) &= \mathrm{LV}_{entry}(4)\\
\mathrm{LV}_{exit}(4) &= \mathrm{LV}_{entry}(5) \cup \mathrm{LV}_{entry}(6)\\
\mathrm{LV}_{exit}(5) &= \mathrm{LV}_{entry}(7)\\
\mathrm{LV}_{exit}(6) &= \mathrm{LV}_{entry}(7)\\
\mathrm{LV}_{exit}(7) &= \emptyset
\end{aligned}
$$

LV Example: Solutions

ℓ	$\mathsf{LV}_{entry}(\ell)$	$\mathsf{LV}_{exit}(\ell)$
1	$\emptyset$	$\emptyset$
2	$\emptyset$	$\{y\}$
3	$\{y\}$	$\{x, y\}$
4	$\{x, y\}$	$\{y\}$
5	$\{y\}$	$\{z\}$
6	$\{y\}$	$\{z\}$
7	$\{z\}$	$\emptyset$

LV Example: Solutions

ℓ	$\mathsf{LV}_{entry}(\ell)$	$\mathsf{LV}_{exit}(\ell)$
1	$\emptyset$	$\emptyset$
2	$\emptyset$	$\{y\}$
3	$\{y\}$	$\{x, y\}$
4	$\{x, y\}$	$\{y\}$
5	$\{y\}$	$\{z\}$
6	$\{y\}$	$\{z\}$
7	$\{z\}$	$\emptyset$

$[\ x := 2\ ]^1;\ [\ y := 4\ ]^2;\ [\ x := 1\ ]^3;$
$(\textbf{if}\ \ [y > x]^4\ \textbf{then}\ \ [\ z := y\ ]^5\ \textbf{else}\ \ [\ z := y * y\ ]^6\ );$
$[\ x := z\ ]^7$

LV Variations

Some authors assume that the variables of interest are output at the end of the program.

LV Variations

Some authors assume that the variables of interest are output at the end of the program.

In that case $\mathrm{LV}_{exit}(7)$ should be $\{x, y, z\}$ which means that $\mathrm{LV}_{entry}(7)$, $\mathrm{LV}_{exit}(5)$ and $\mathrm{LV}_{exit}(6)$ should all be $\{y, z\}$.