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:

- 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.*

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:

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

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

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

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

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

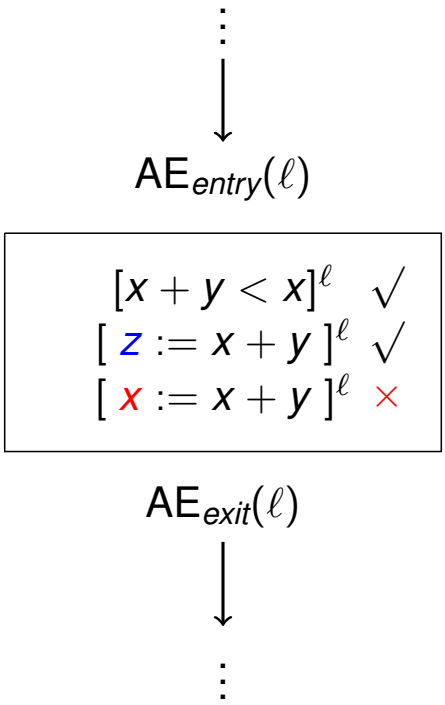
$$\mathsf{AE}_{exit} : \mathbf{Lab}_\star \to \mathcal{P}(\mathbf{AExp}_\star)$$

AE Auxiliary Functions

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

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

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



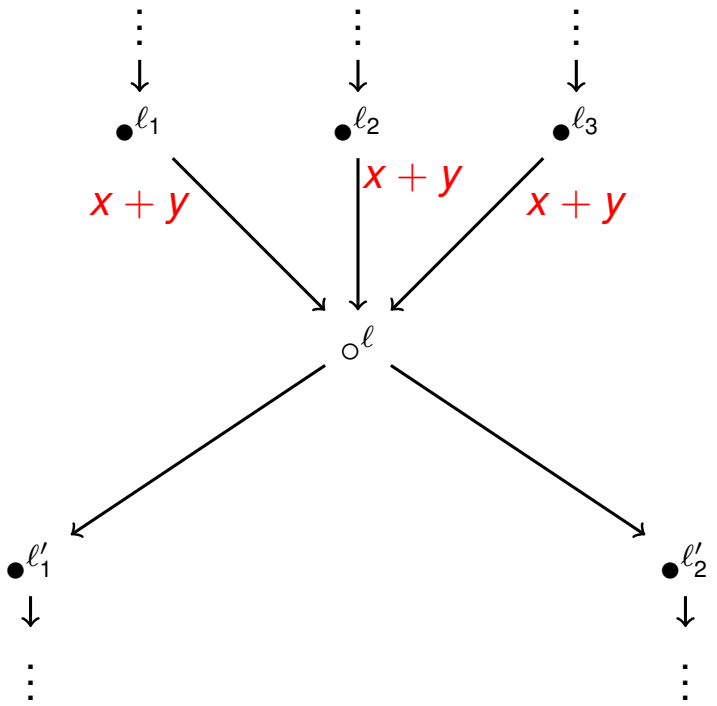
Whenever a variable x in an expression gets a new value the expression becomes unavailable.

AE Equation Schemes

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

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

AE Global Collection



We push information “forward in time”.

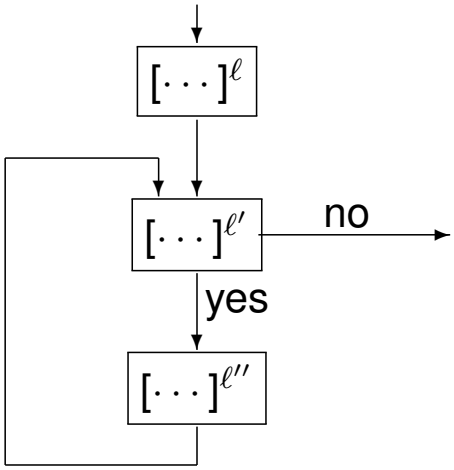
Largest Solution

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

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

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

Obtaining Solutions



After some simplification, we find that:

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

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

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

$$
\begin{array}{lcl}
\mathsf{AE}_{exit}(1) & = & \mathsf{AE}_{entry}(1) \cup \{a + b\} \\
\mathsf{AE}_{exit}(2) & = & \mathsf{AE}_{entry}(2) \cup \{a * b\} \\
\mathsf{AE}_{exit}(3) & = & \mathsf{AE}_{entry}(3) \cup \{a + b\} \\
\mathsf{AE}_{exit}(4) & = & \mathsf{AE}_{entry}(4) \setminus \{a + b, a * b, a + 1\} \\
\mathsf{AE}_{exit}(5) & = & \mathsf{AE}_{entry}(5) \cup \{a + b\}
\end{array}
$$

AE Example: Equations

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

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

AE Example: Equations

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

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

AE Example: Solutions

ℓ	$\mathrm{AE}_{entry}(\ell)$	$\mathrm{AE}_{exit}(\ell)$
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.*

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:

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

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

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

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

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

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

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

RD Auxiliary Functions

$$\begin{array}{rcl} kill_{\mathsf{RD}}([\, x := a \,]^\ell) & = & \{(x, ?)\} \cup \{(x, \ell') \mid \\ & & [B]^{\ell'} \text{ a “definition” of } x \text{ in } S_\star\} \\ kill_{\mathsf{RD}}([\, \mathbf{skip} \,]^\ell) & = & \emptyset \\ kill_{\mathsf{RD}}([b]^\ell) & = & \emptyset \end{array}$$

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

RD Equation Schemes

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

$$\mathrm{RD}_{exit}(\ell) = (\mathrm{RD}_{entry}(\ell) \setminus kill_{\mathrm{RD}}([B]^\ell)) \cup gen_{\mathrm{RD}}([B]^\ell) \quad \text{where } [B]^\ell \in blocks(S_\star)$$

Smallest Solution

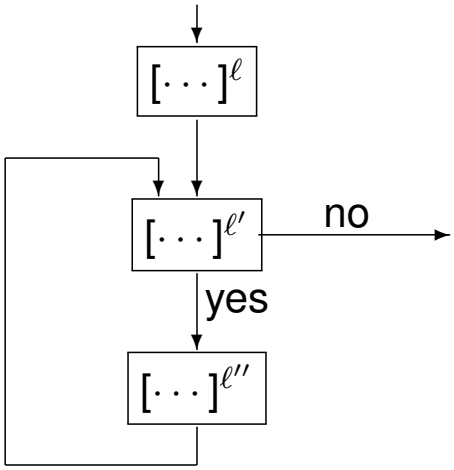
Similar to before, this is a *forward analysis* but we are interested in the *smallest* sets satisfying the equation for RD_{entry}.

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

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

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

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 $\mathsf{RD}_{entry}(init(S_\star)) = \emptyset$ rather than $\mathsf{RD}_{entry}(init(S_\star)) = \{((x, ?) \mid x \in FV(S_\star)\}$.

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

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

ℓ	$kill_{\mathrm{RD}}(\ell)$	$gen_{\mathrm{RD}}(\ell)$
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

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

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

RD Example: Equations

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

$$
\begin{array}{lcl}
RD_{entry}(1) & = & \{(x,?),(y,?)\} \\
RD_{entry}(2) & = & RD_{exit}(1) \\
RD_{entry}(3) & = & RD_{exit}(2) \cup RD_{exit}(5) \\
RD_{entry}(4) & = & RD_{exit}(3) \\
RD_{entry}(5) & = & RD_{exit}(4)
\end{array}
$$

RD Example: Equations

$$
\begin{array}{lcl}
RD_{entry}(1) & = & \{(x,?),(y,?)\} \\
RD_{entry}(2) & = & RD_{exit}(1) \\
RD_{entry}(3) & = & RD_{exit}(2) \cup RD_{exit}(5) \\
RD_{entry}(4) & = & RD_{exit}(3) \\
RD_{entry}(5) & = & RD_{exit}(4)
\end{array}
$$

$$
\begin{array}{lcl}
RD_{exit}(1) & = & (RD_{entry}(1) \setminus \{(x,?),(x,1),(x,5)\}) \cup \{(x,1)\} \\
RD_{exit}(2) & = & (RD_{entry}(2) \setminus \{(y,?),(y,2),(y,4)\}) \cup \{(y,2)\} \\
RD_{exit}(3) & = & RD_{entry}(3) \\
RD_{exit}(4) & = & (RD_{entry}(4) \setminus \{(y,?),(y,2),(y,4)\}) \cup \{(y,4)\} \\
RD_{exit}(5) & = & (RD_{entry}(5) \setminus \{(x,?),(x,1),(x,5)\}) \cup \{(x,5)\}
\end{array}
$$

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	$\{(x,1),(y,4),(x,5)\}$	$\{(y,4),(x,5)\}$

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

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:

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

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

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

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

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

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

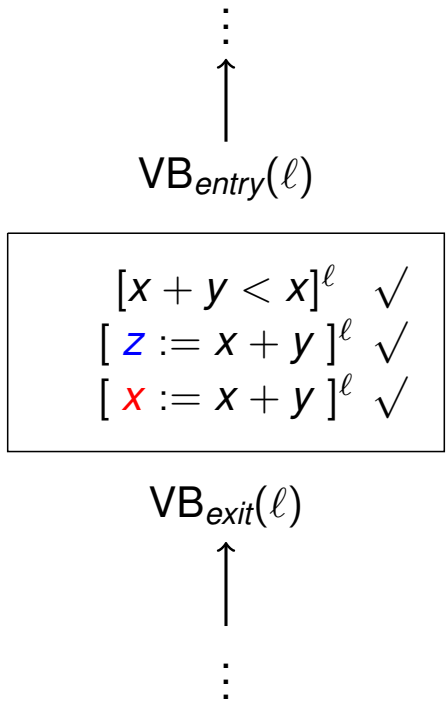
The analysis is a *backward analysis* and we are interested in the *largest* sets satisfying the equation for VB_{exit}.

VB Auxiliary Functions

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

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

VB Local Change



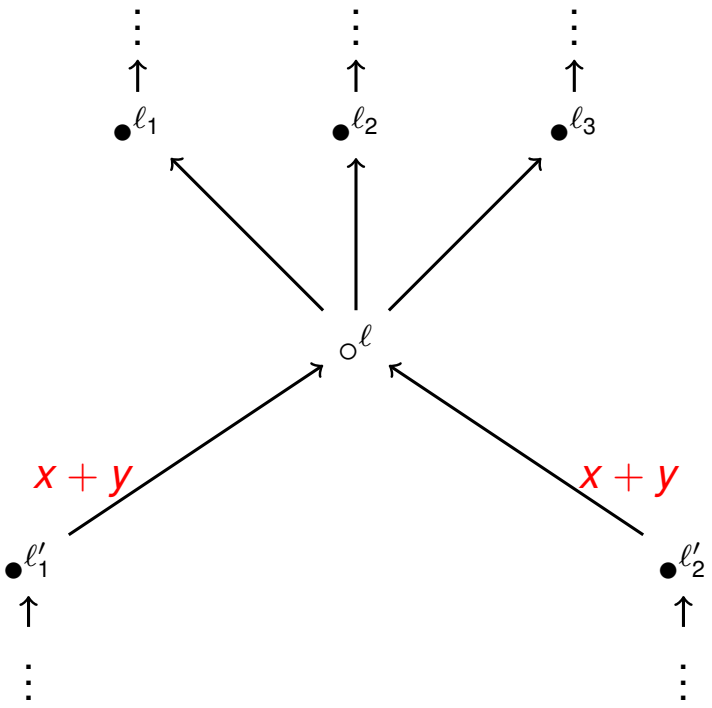
Whenever a variable x in an expression gets a new value it does not help us if it was evaluated before.

VB Equation Schemes

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

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

VB Global Collection



We need to go “back in time”.

VB Example

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

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

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

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

VB Example: Equations

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

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

VB Example: Equations

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

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

VB Example: 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	$\{a-b\}$	$\emptyset$

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

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:

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

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

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

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

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

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

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

LV Auxiliary Functions

$$
\begin{array}{rcl}
kill_{\mathrm{LV}}([\ x := a\ ]^{\ell}) & = & \{x\} \\
kill_{\mathrm{LV}}([\ \textbf{skip}\ ]^{\ell}) & = & \emptyset \\
kill_{\mathrm{LV}}([b]^{\ell}) & = & \emptyset
\end{array}
$$

$$
\begin{array}{rcl}
gen_{\mathrm{LV}}([\ x := a\ ]^{\ell}) & = & FV(a) \\
gen_{\mathrm{LV}}([\ \textbf{skip}\ ]^{\ell}) & = & \emptyset \\
gen_{\mathrm{LV}}([b]^{\ell}) & = & FV(b)
\end{array}
$$

LV Equation Schemes

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

$$
\begin{array}{rcl}
\mathrm{LV}_{entry}(\ell) & = & (\mathrm{LV}_{exit}(\ell) \setminus kill_{\mathrm{LV}}([B]^{\ell}) \cup gen_{\mathrm{LV}}([B]^{\ell}) \\
 & & \text{where } [B]^{\ell} \in blocks(S_\star)
\end{array}
$$

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

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

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

LV Example: Equations

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

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

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	$\{z\}$	$\emptyset$

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

LV Variations

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

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 $\{y, z\}$.