Remark:

If $f : \mathbb{D}_1 \longrightarrow \mathbb{D}_2$ is distributive, then also monotonic :-)

Remark:

If $f : \mathbb{D}_1 \longrightarrow \mathbb{D}_2$ is distributive, then also monotonic :-)

Obviously: $a \sqsubseteq b$ iff $a \sqcup b = b$.

Remark:

If $f : \mathbb{D}_1 \to \mathbb{D}_2$ is distributive, then also monotonic :-)

Obviously: $a \sqsubseteq b$ iff $a \sqcup b = b$.

From that follows:

$$\begin{array}{rrcl} & f\,b & = & f\,(a \sqcup b) \\ & & = & f\,a \sqcup f\,b \\ \Longrightarrow & f\,a & \sqsubseteq & f\,b \end{array}$$

:-)

Assumption: all v are reachable from $start$.

Assumption: all v are reachable from *start* .

Then:

Theorem

Kildall 1972

If all effects of edges $[\![k]\!]^\sharp$ are distributive, then: $\mathcal{I}^*[v] = \mathcal{I}[v]$

for all v .

Gary A. Kildall (1942-1994).

Has developed the operating system CP/M and GUIs for PCs.

Assumption: all v are reachable from *start* .
Then:

Theorem Kildall 1972

If all effects of edges $[\![k]\!]^\sharp$ are distributive, then: $\mathcal{I}^*[v] = \mathcal{I}[v]$
for all v .

Assumption: all v are reachable from *start* .
Then:

Theorem

Kildall 1972

If all effects of edges $[\![k]\!]^\sharp$ are distributive, then: $\mathcal{I}^*[v] = \mathcal{I}[v]$ for all v .

Proof:

It suffices to prove that $\mathcal{I}^*$ is a solution :-)

For this, we show that $\mathcal{I}^*$ satisfies all constraints :-))

(1) We prove for *start* :

$$\begin{array}{lll}
\mathcal{I}^*[\mathit{start}] & = & \bigsqcup\{[\![\pi]\!]^\sharp\, d_0 \mid \pi : \mathit{start} \longrightarrow^* \mathit{start}\} \\
& \sqsupseteq & [\![\epsilon]\!]^\sharp\, d_0 \\
& \sqsupseteq & d_0 \qquad \text{:-)}
\end{array}$$

(1) We prove for $start$:

$$\begin{array}{lll} \mathcal{I}^*[start] & = & \bigsqcup\{[\![\pi]\!]^\sharp\, d_0 \mid \pi : start \rightarrow^* start\} \\ & \sqsupseteq & [\![\epsilon]\!]^\sharp\, d_0 \\ & \sqsupseteq & d_0 \qquad \text{:-)} \end{array}$$

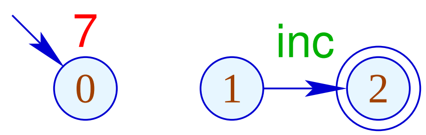
(2) For every $k = (u, \_, v)$ we prove:

$$\begin{array}{lll} \mathcal{I}^*[v] & = & \bigsqcup\{[\![\pi]\!]^\sharp\, d_0 \mid \pi : start \rightarrow^* v\} \\ & \sqsupseteq & \bigsqcup\{[\![\pi' k]\!]^\sharp\, d_0 \mid \pi' : start \rightarrow^* u\} \\ & = & \bigsqcup\{[\![k]\!]^\sharp\, ([\![\pi']\!]^\sharp\, d_0) \mid \pi' : start \rightarrow^* u\} \\ & = & [\![k]\!]^\sharp\, (\bigsqcup\{[\![\pi']\!]^\sharp\, d_0 \mid \pi' : start \rightarrow^* u\}) \\ & = & [\![k]\!]^\sharp\, (\mathcal{I}^*[u]) \end{array}$$

since $\{\pi' \mid \pi' : start \rightarrow^* u\}$ is non-empty :-)

Warning:

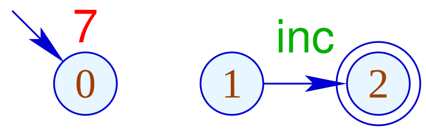
- **Reachability** of all program points cannot be abandoned!
  Consider:



where $\mathbb{D} = \mathbb{N} \cup \{\infty\}$

Warning:

- **Reachability** of all program points cannot be abandoned!
  Consider:

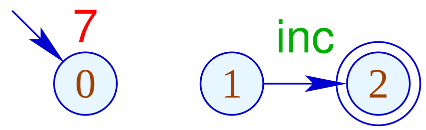
  where $\mathbb{D} = \mathbb{N} \cup \{\infty\}$

  Then:

$$\begin{aligned} \mathcal{I}[2] &= \mathsf{inc}\,0 &= 1 \\ \mathcal{I}^*[2] &= \bigsqcup \emptyset &= 0 \end{aligned}$$

Warning:

- Reachability of all program points cannot be abandoned!
  Consider:

  where $\mathbb{D} = \mathbb{N} \cup \{\infty\}$

  Then:

$$
\begin{array}{lllll}
\mathcal{I}[2] & = & \mathsf{inc}\,0 & = & 1 \\
\mathcal{I}^*[2] & = & \bigsqcup \emptyset & = & 0
\end{array}
$$

- Unreachable program points can always be thrown away :-)

Summary and Application:

→ The effects of edges of the analysis of availability of expressions are distributive:

$$
\begin{aligned}
(a \cup (x_1 \cap x_2)) \backslash b &= ((a \cup x_1) \cap (a \cup x_2)) \backslash b \\
&= ((a \cup x_1) \backslash b) \cap ((a \cup x_2) \backslash b)
\end{aligned}
$$

Summary and Application:

→ The effects of edges of the analysis of availability of expressions are distributive:

$$\begin{array}{lll} (a \cup (x_1 \cap x_2)) \backslash b & = & ((a \cup x_1) \cap (a \cup x_2)) \backslash b \\ & = & ((a \cup x_1) \backslash b) \cap ((a \cup x_2) \backslash b) \end{array}$$

→ If all effects of edges are distributive, then the MOP can be computed by means of the constraint system and RR-iteration. :-)

Summary and Application:

$\longrightarrow$ The effects of edges of the analysis of availability of expressions are distributive:

$$\begin{array}{lll}(a \cup (x_1 \cap x_2))\backslash b & = & ((a \cup x_1) \cap (a \cup x_2))\backslash b \\ & = & ((a \cup x_1)\backslash b) \cap ((a \cup x_2)\backslash b)\end{array}$$

$\longrightarrow$ If all effects of edges are distributive, then the MOP can be computed by means of the constraint system and RR-iteration. :-)

$\longrightarrow$ If not all effects of edges are distributive, then RR-iteration for the constraint system at least returns a safe upper bound to the MOP :-)

1.2 Removing Assignments to Dead Variables

Example:

$$
\begin{array}{ll}
1: & x = y + 2; \\
2: & y = 5; \\
3: & x = y + 3;
\end{array}
$$

The value of x at program points $1, 2$ is over-written before it can be used.

Therefore, we call the variable x dead at these program points :-)

Note:

→ Assignments to dead variables can be removed ;-)

→ Such inefficiencies may originate from other transformations.

Note:

- $\longrightarrow$ Assignments to dead variables can be removed ;-)
- $\longrightarrow$ Such inefficiencies may originate from other transformations.

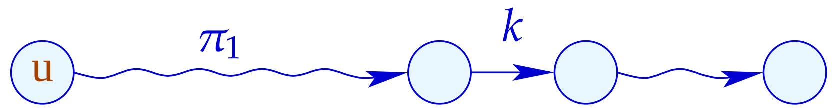
Formal Definition:

The variable x is called live at u along the path π starting at u relative to a set X of variables either:

if $x \in X$ and π does not contain a definition of x; or:

if π can be decomposed into: $\pi = \pi_1 \, k \, \pi_2$ such that:

- k is a use of x ; and
- π_1 does not contain a definition of x.

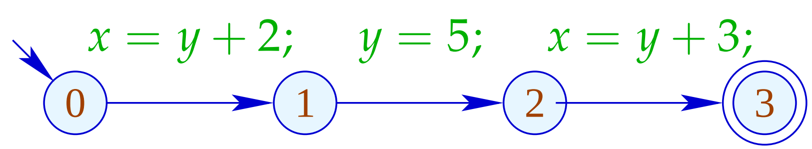


Thereby, the set of all defined or used variables at an edge $k = (\_, lab, \_)$ is defined by:

lab	used	defined
$;$	$\emptyset$	$\emptyset$
$\mathrm{Pos}\,(e)$	$Vars\,(e)$	$\emptyset$
$\mathrm{Neg}\,(e)$	$Vars\,(e)$	$\emptyset$
$x = e;$	$Vars\,(e)$	$\{x\}$
$x = M[e];$	$Vars\,(e)$	$\{x\}$
$M[e_1] = e_2;$	$Vars\,(e_1) \cup Vars\,(e_2)$	$\emptyset$

A variable x which is not live at u along π (relative to X) is called dead at u along π (relative to X).

Example:



where $X = \emptyset$. Then we observe:

	live	dead
0	$\{y\}$	$\{x\}$
1	$\emptyset$	$\{x, y\}$
2	$\{y\}$	$\{x\}$
3	$\emptyset$	$\{x, y\}$

The variable x is live at u (relative to X) if x is live at u along some path to the exit (relative to X). Otherwise, x is called dead at u (relative to X).

The variable x is live at u (relative to X) if x is live at u along some path to the exit (relative to X). Otherwise, x is called dead at u (relative to X).

Question:

How can the sets of all dead/live variables be computed for every u ???

The variable x is live at u (relative to X) if x is live at u along some path to the exit (relative to X). Otherwise, x is called dead at u (relative to X).

Question:

How can the sets of all dead/live variables be computed for every u ???

Idea:

For every edge $k = (u, \_, v)$, define a function $[\![k]\!]^\sharp$ which transforms the set of variables which are live at v into the set of variables which are live at u ...

Let $\quad \mathbb{L} = 2^{Vars}$.

For $\quad k = (\_, lab, \_)$, define $\quad [\![k]\!]^\sharp = [\![lab]\!]^\sharp \quad$ by:

$$
\begin{array}{lll}
[\![;]\!]^\sharp\, L & = & L \\
[\![\mathrm{Pos}(e)]\!]^\sharp\, L & = & [\![\mathrm{Neg}(e)]\!]^\sharp\, L \quad = \quad L \cup Vars(e) \\
[\![x = e;]\!]^\sharp\, L & = & (L \backslash \{x\}) \cup Vars(e) \\
[\![x = M[e];]\!]^\sharp\, L & = & (L \backslash \{x\}) \cup Vars(e) \\
[\![M[e_1] = e_2;]\!]^\sharp\, L & = & L \cup Vars(e_1) \cup Vars(e_2)
\end{array}
$$

Let $\mathbb{L} = 2^{Vars}$.

For $k = (\_, lab, \_)$, define $[\![k]\!]^\sharp = [\![lab]\!]^\sharp$ by:

$$
\begin{array}{lcl}
[\![;]\!]^\sharp\, L & = & L \\
[\![\mathrm{Pos}(e)]\!]^\sharp\, L & = & [\![\mathrm{Neg}(e)]\!]^\sharp\, L \quad = \quad L \cup Vars(e) \\
[\![x = e;]\!]^\sharp\, L & = & (L \backslash \{x\}) \cup Vars(e) \\
[\![x = M[e];]\!]^\sharp\, L & = & (L \backslash \{x\}) \cup Vars(e) \\
[\![M[e_1] = e_2;]\!]^\sharp\, L & = & L \cup Vars(e_1) \cup Vars(e_2)
\end{array}
$$

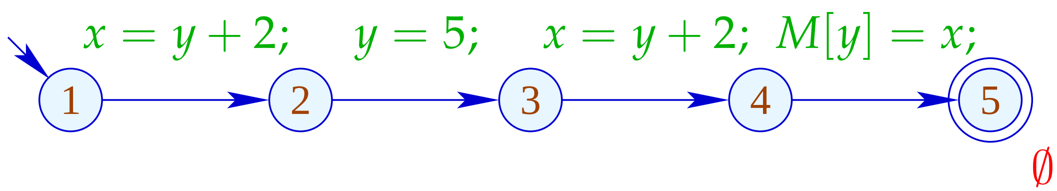
$[\![k]\!]^\sharp$ can again be composed to the effects of $[\![\pi]\!]^\sharp$ of paths $\pi = k_1 \ldots k_r$ by:

$$[\![\pi]\!]^\sharp = [\![k_1]\!]^\sharp \circ \ldots \circ [\![k_r]\!]^\sharp$$

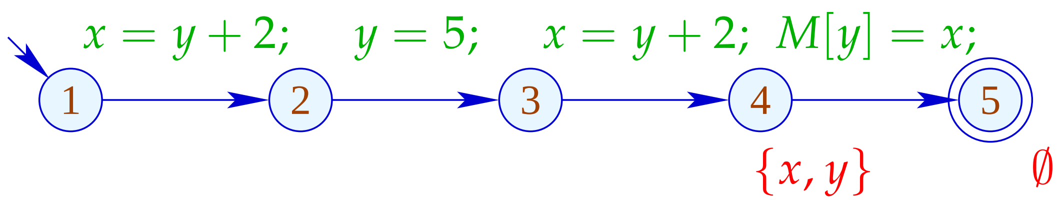
We verify that these definitions are meaningful :-)

$$1 \xrightarrow{x = y + 2;} 2 \xrightarrow{y = 5;} 3 \xrightarrow{x = y + 2;} 4 \xrightarrow{M[y] = x;} 5$$

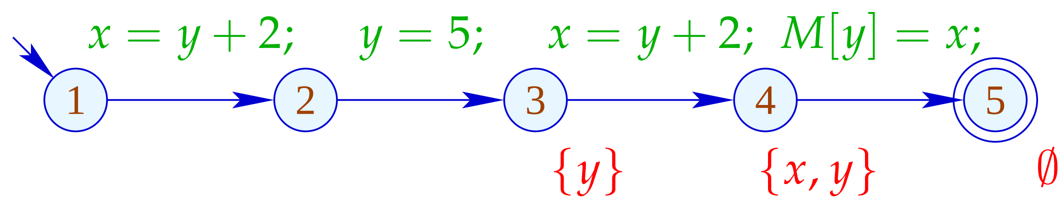
We verify that these definitions are meaningful :-)



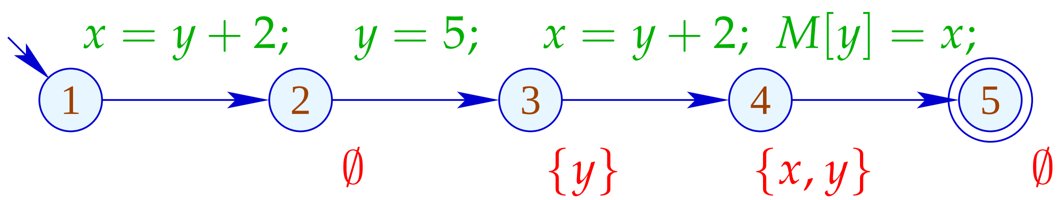
We verify that these definitions are meaningful :-)



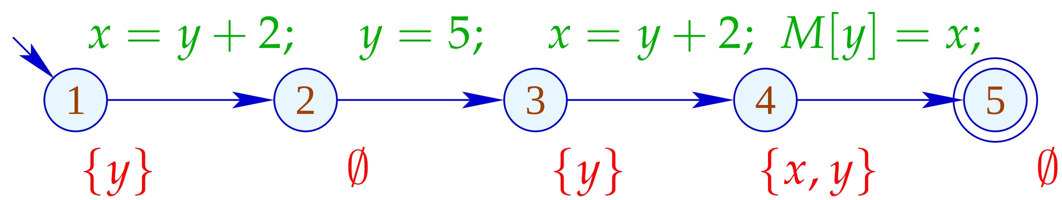
We verify that these definitions are meaningful :-)



We verify that these definitions are meaningful :-)



We verify that these definitions are meaningful :-)



The set of variables which are live at u then is given by:

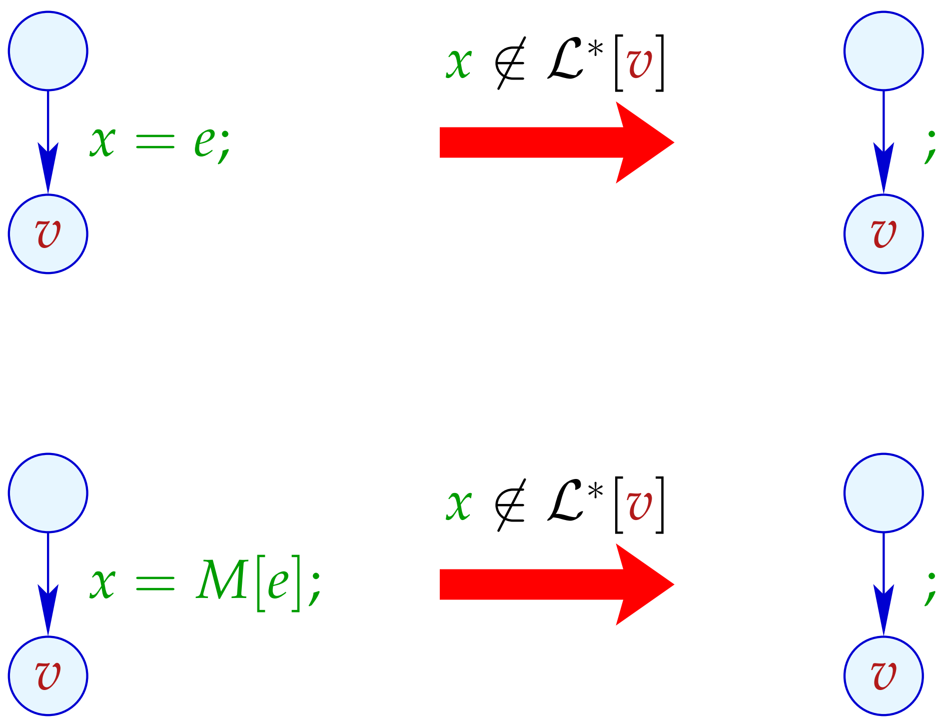
$$\mathcal{L}^*[u] = \bigcup\{[\![\pi]\!]^\sharp X \mid \pi : u \to^* stop\}$$

... literally:

- The paths start in u :-)

  $\Longrightarrow$ As partial ordering for $\mathbb{L}$ we use $\sqsubseteq = \subseteq$.
- The set of variables which are live at program exit is given by the set X :-)

Transformation 2:



Correctness Proof:

- $\longrightarrow$ Correctness of the effects of edges: If L is the set of variables which are live at the exit of the path π , then $[\![\pi]\!]^\sharp\, L$ is the set of variables which are live at the beginning of π :-)
- $\longrightarrow$ Correctness of the transformation along a path: If the value of a variable is accessed, this variable is necessarily live. The value of dead variables thus is irrelevant :-)
- $\longrightarrow$ Correctness of the transformation: In any execution of the transformed programs, the live variables always receive the same values :-))

Computation of the sets $\mathcal{L}^*[u]$:

(1) Collecting constraints:

$$\begin{array}{lll} \mathcal{L}[stop] & \supseteq & X \\ \mathcal{L}[u] & \supseteq & [\![k]\!]^\sharp\,(\mathcal{L}[v]) \qquad k = (u, \_, v) \quad \text{edge} \end{array}$$

(2) Solving the constraint system by means of RR iteration.
Since $\mathbb{L}$ is finite, the iteration will terminate :-)

(3) If the exit is (formally) reachable from every program point, then the smallest solution $\mathcal{L}$ of the constraint system equals $\mathcal{L}^*$ since all $[\![k]\!]^\sharp$ are distributive :-))

Computation of the sets $\mathcal{L}^*[u]$:

(1) Collecting constraints:

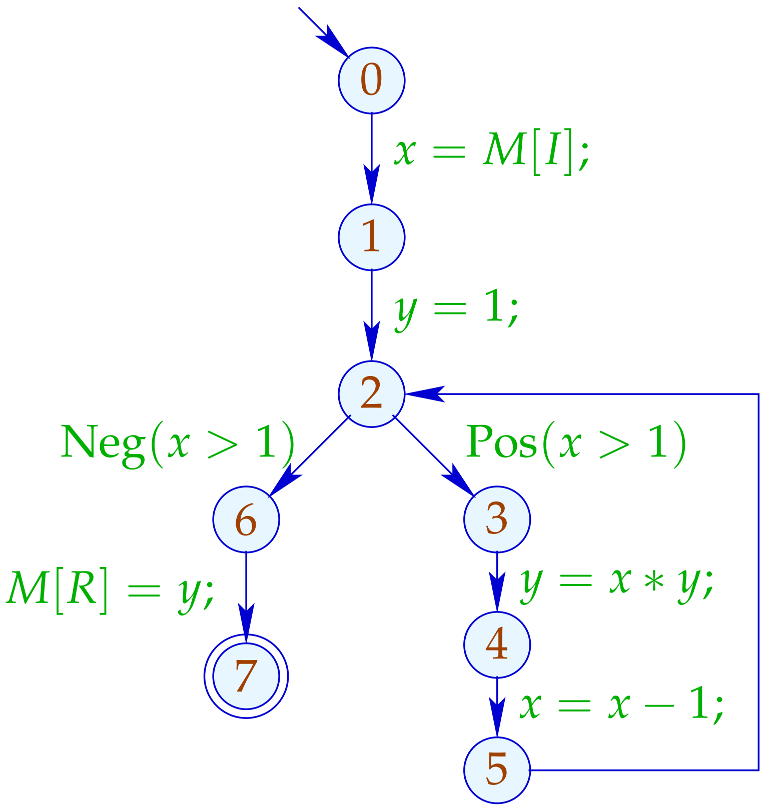
$$\begin{array}{lll} \mathcal{L}[stop] & \supseteq & X \\ \mathcal{L}[u] & \supseteq & [\![k]\!]^\sharp\,(\mathcal{L}[v]) \qquad k = (u, \_, v) \quad \text{edge} \end{array}$$

(2) Solving the constraint system by means of RR iteration.
Since $\mathbb{L}$ is finite, the iteration will terminate :-)

(3) If the exit is (formally) reachable from every program point, then the smallest solution $\mathcal{L}$ of the constraint system equals $\mathcal{L}^*$ since all $[\![k]\!]^\sharp$ are distributive :-))

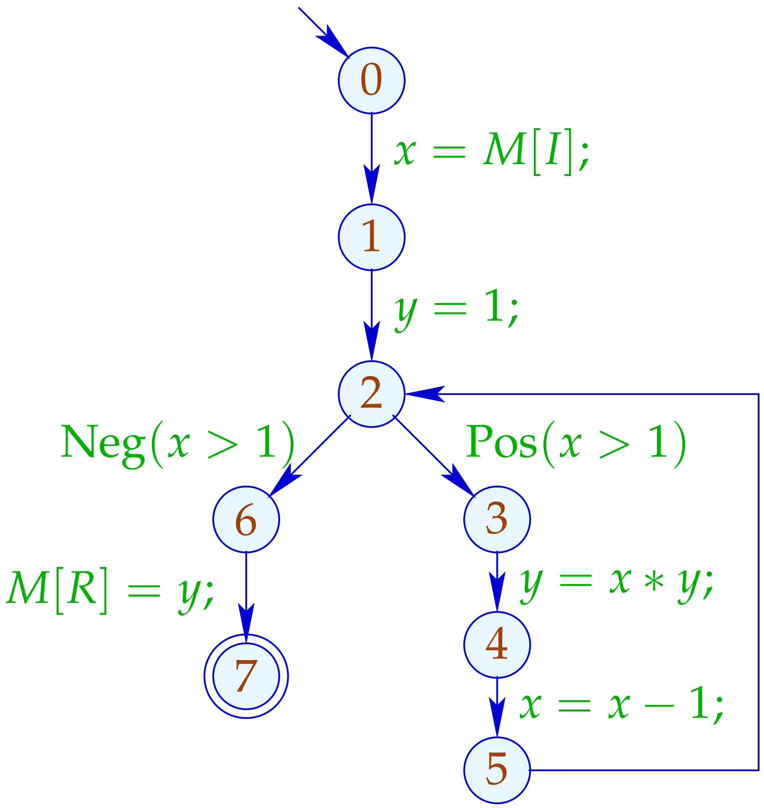
Warning: The information is propagated backwards !!!

Example:



$$
\begin{array}{lll}
\mathcal{L}[0] & \supseteq & (\mathcal{L}[1] \backslash \{x\}) \cup \{I\} \\
\mathcal{L}[1] & \supseteq & \mathcal{L}[2] \backslash \{y\} \\
\mathcal{L}[2] & \supseteq & (\mathcal{L}[6] \cup \{x\}) \cup (\mathcal{L}[3] \cup \{x\}) \\
\mathcal{L}[3] & \supseteq & (\mathcal{L}[4] \backslash \{y\}) \cup \{x, y\} \\
\mathcal{L}[4] & \supseteq & (\mathcal{L}[5] \backslash \{x\}) \cup \{x\} \\
\mathcal{L}[5] & \supseteq & \mathcal{L}[2] \\
\mathcal{L}[6] & \supseteq & \mathcal{L}[7] \cup \{y, R\} \\
\mathcal{L}[7] & \supseteq & \emptyset
\end{array}
$$

Example:

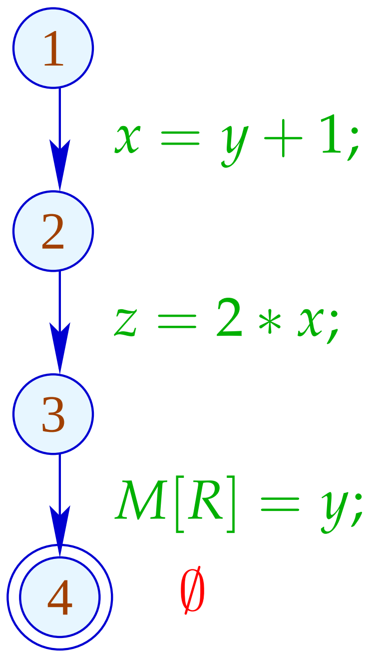


	1	2
7	$\emptyset$	
6	$\{y, R\}$	
2	$\{x, y, R\}$	dito
5	$\{x, y, R\}$	
4	$\{x, y, R\}$	
3	$\{x, y, R\}$	
1	$\{x, R\}$	
0	$\{I, R\}$	

The left-hand side of no assignment is dead :-)

Warning:

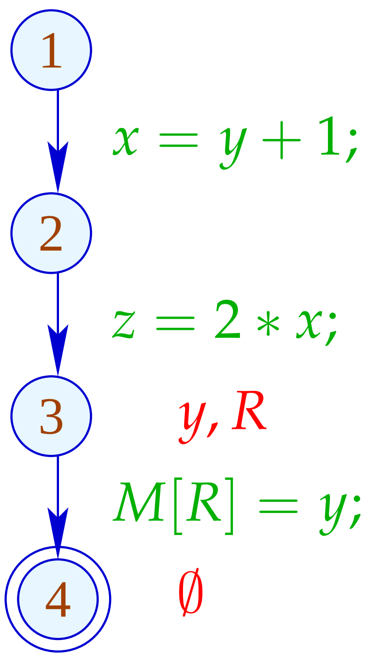
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is dead :-)

Warning:

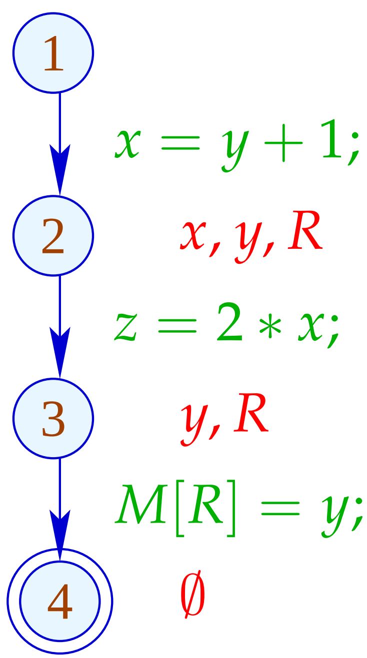
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is dead :-)

Warning:

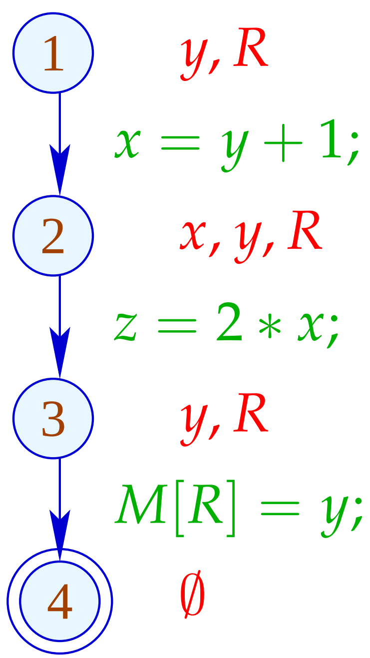
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is dead :-)

Warning:

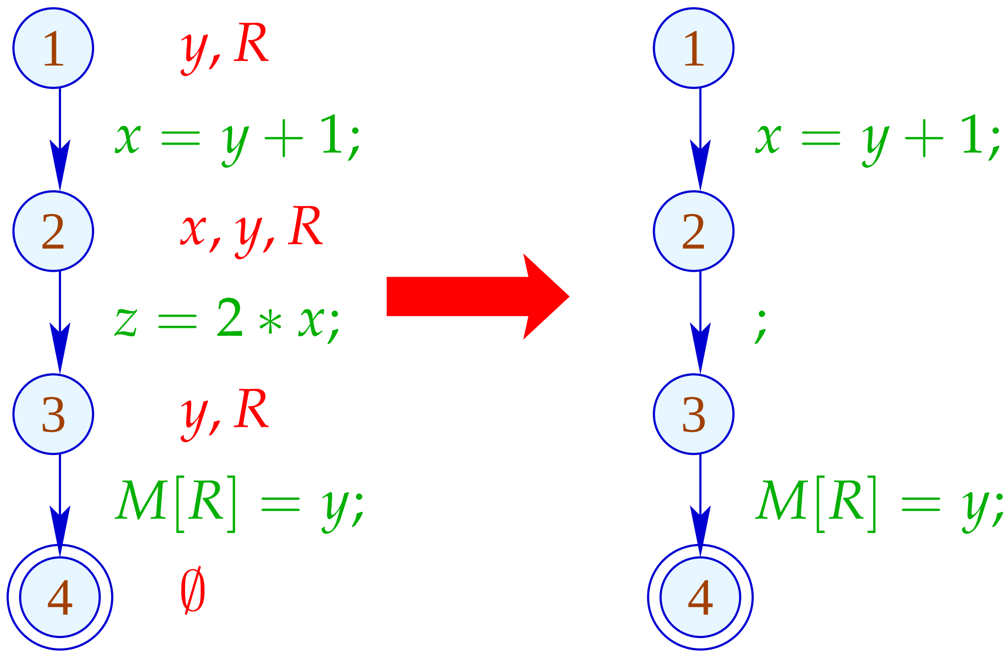
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is dead :-)

Warning:

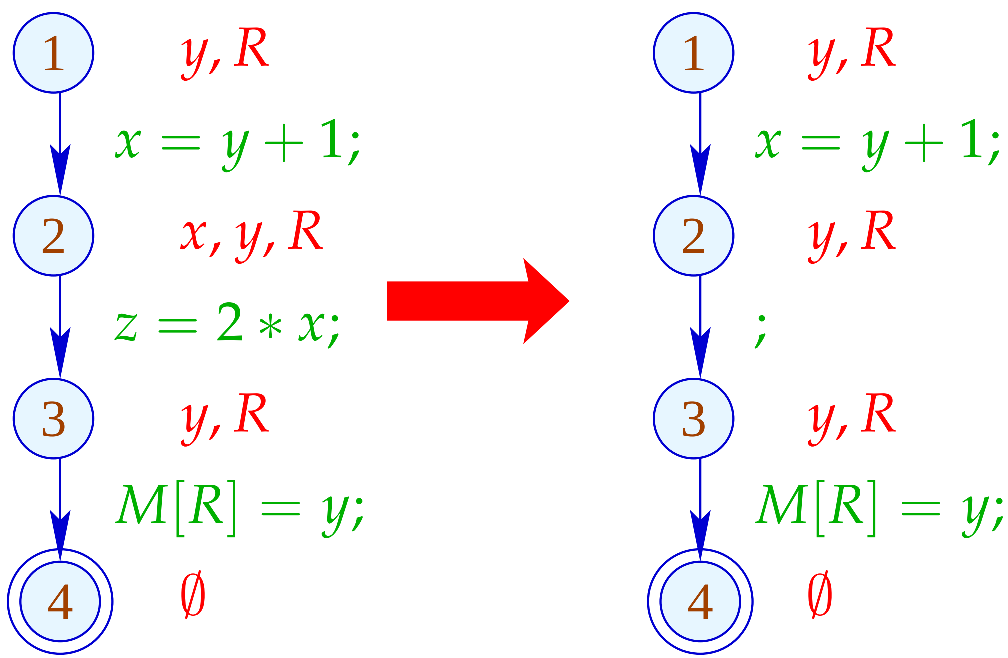
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is dead :-)

Warning:

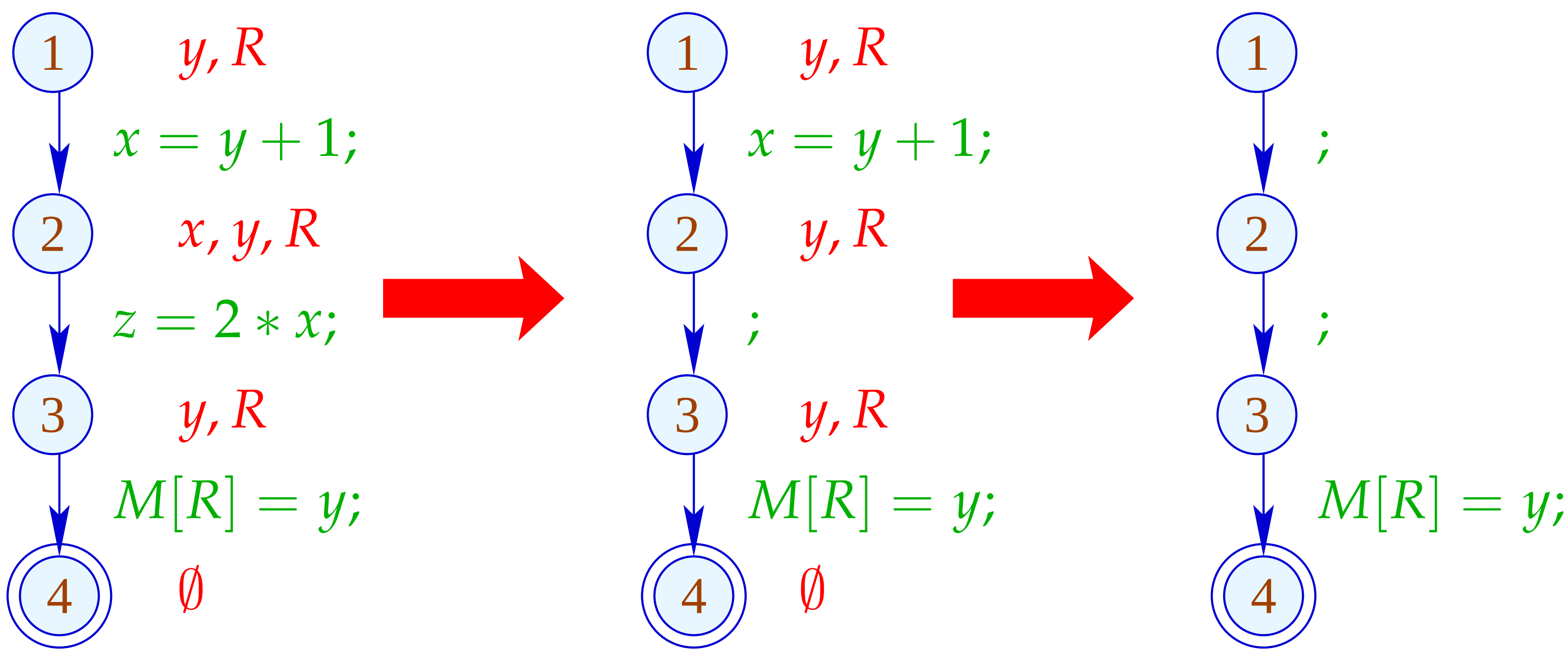
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is dead :-)

Warning:

Removal of assignments to dead variables may kill further variables:



Re-analyzing the program is inconvenient :-(

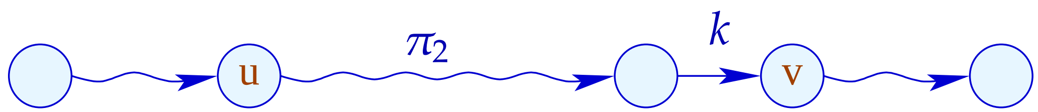
Idea: Analyze true liveness!

x is called truely live at u along a path π (relative to X), either

if $x \in X$, π does not contain a definition of x; or

if π can be decomposed into $\pi = \pi_1\, k\, \pi_2$ such that:

- k is a true use of x ;
- π_1 does not contain any definition of x.



The set of truely used variables at an edge $k = (\_, lab, v)$ is defined as:

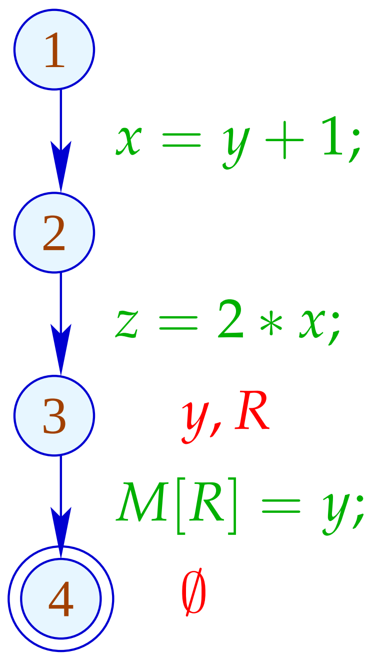
lab	truely used
$;$	$\emptyset$
Pos (e)	$Vars\,(e)$
Neg (e)	$Vars\,(e)$
$x = e;$	$Vars\,(e)$ $(*)$
$x = M[e];$	$Vars\,(e)$ $(*)$
$M[e_1] = e_2;$	$Vars(e_1) \cup Vars(e_2)$

$(*)$ – given that x is truely live at v :-)

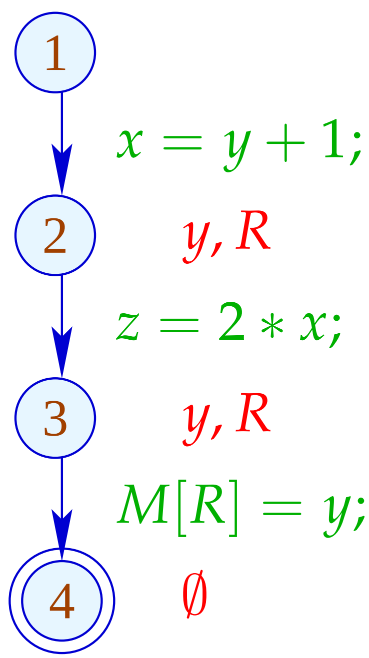
Example:

$$\begin{tikzcd}
1 \arrow[d, "x = y + 1;"] \\
2 \arrow[d, "z = 2 * x;"] \\
3 \arrow[d, "{M[R] = y;}"] \\
4 & \emptyset
\end{tikzcd}$$

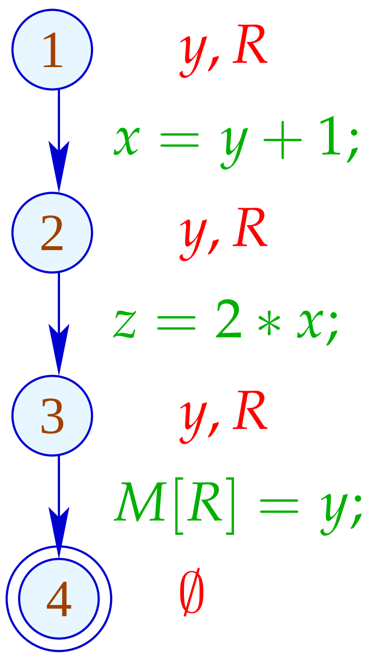
Example:



Example:



Example:



Example: