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

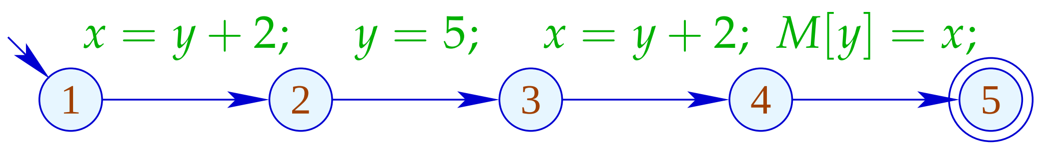
For $k = (\_, lab, \_)$, define $[\![k]\!]^\sharp = [\![lab]\!]^\sharp$ by:

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

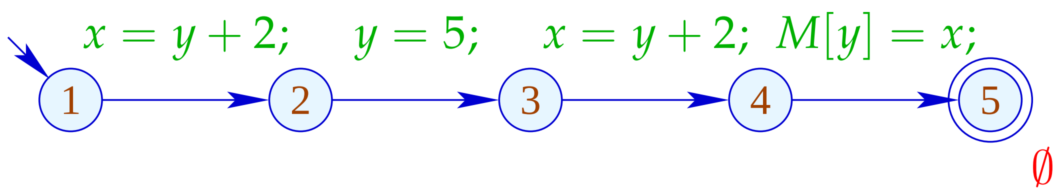
$[\![k]\!]^\sharp$ can again be composed to the effects of $[\![\pi]\!]^\sharp$ of paths $\pi = k_1 \ldots k_r$ by:

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

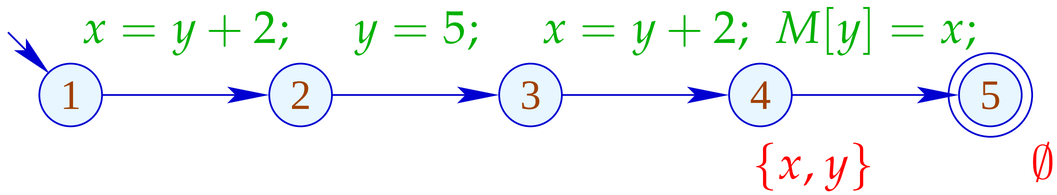
We verify that these definitions are meaningful :-)



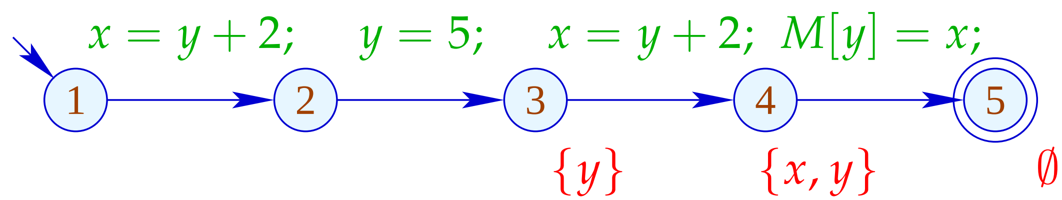
We verify that these definitions are meaningful :-)



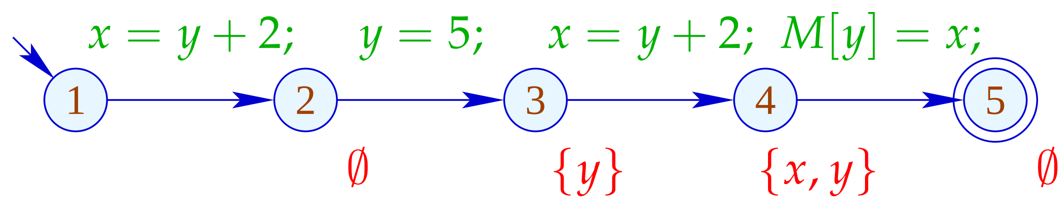
We verify that these definitions are meaningful :-)



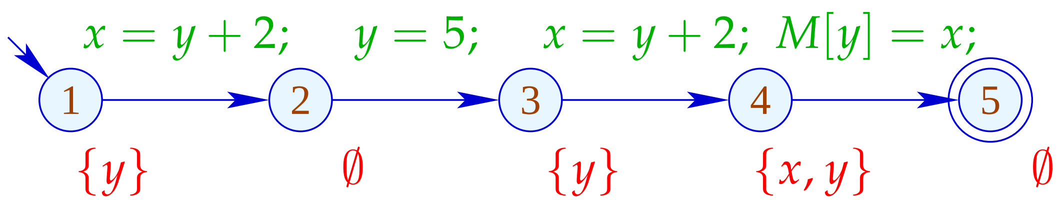
We verify that these definitions are meaningful :-)



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:

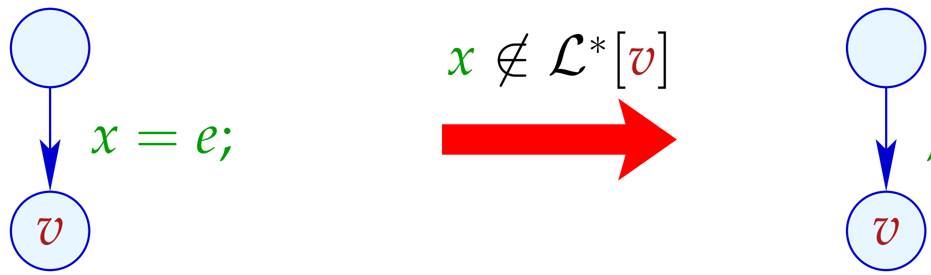
$$\mathcal{L}^*[u] = \bigcup\{[\![\pi]\!]^\sharp X \mid \pi : u \rightarrow^* stop\}$$

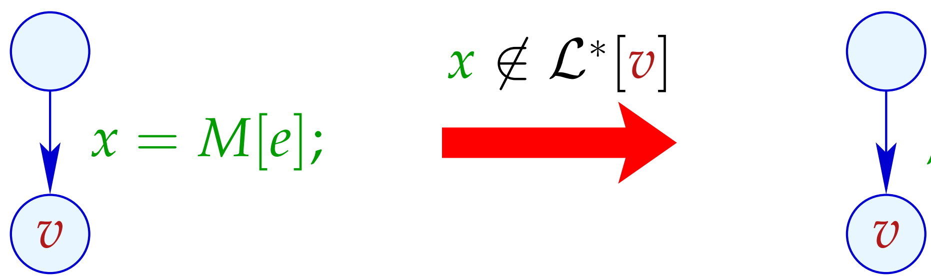
... literally:

- 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:

→ 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 :-)

→ 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 :-)

→ 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:

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

(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:

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

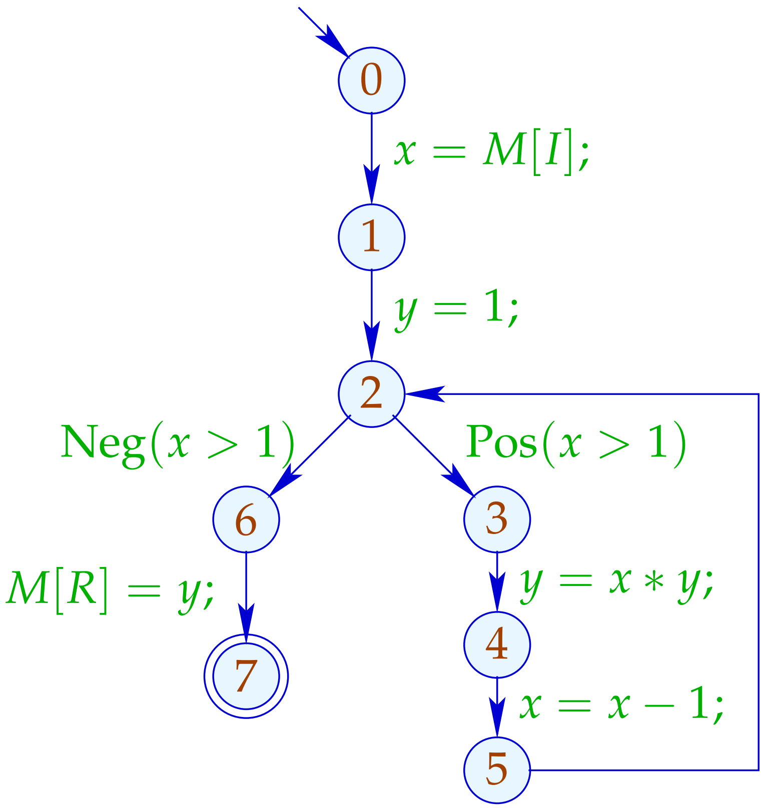
(2) Solving the constraint system by means of RR iteration.

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 :-))

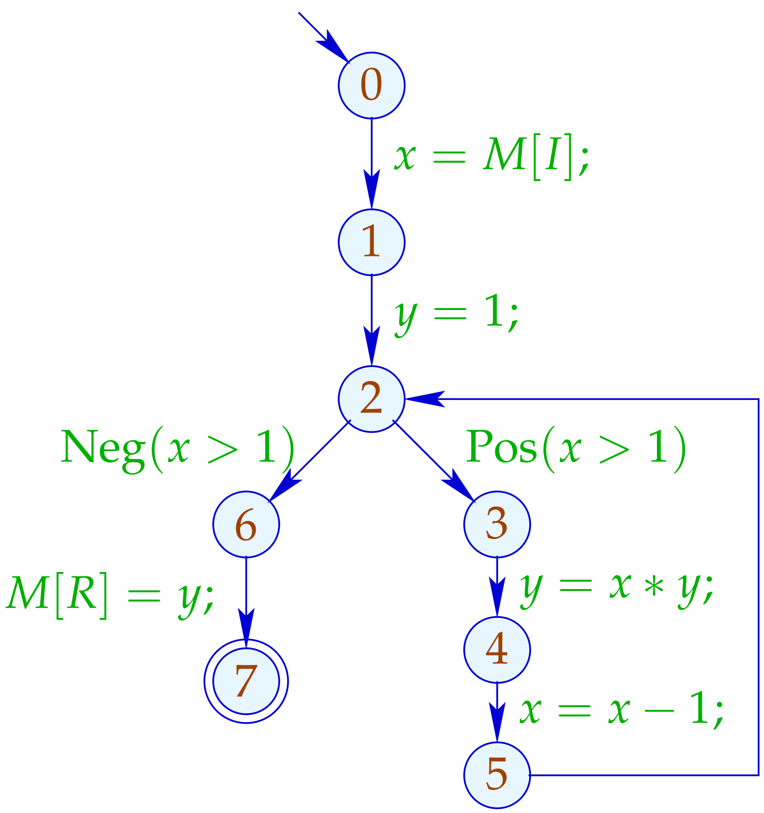
Warning: The information is propagated backwards !!!

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:

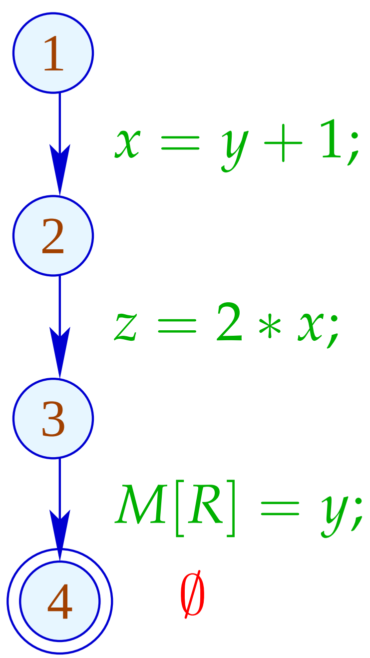


	1	2
7	$\emptyset$	
6	$\{y, R\}$	
2	$\{x, y, R\}$	dito
5	$\{x, y, R\}$	
4	$\{x, y, R\}$	
3	$\{x, y, R\}$	
1	$\{x, R\}$	
0	$\{I, R\}$	

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

Warning:

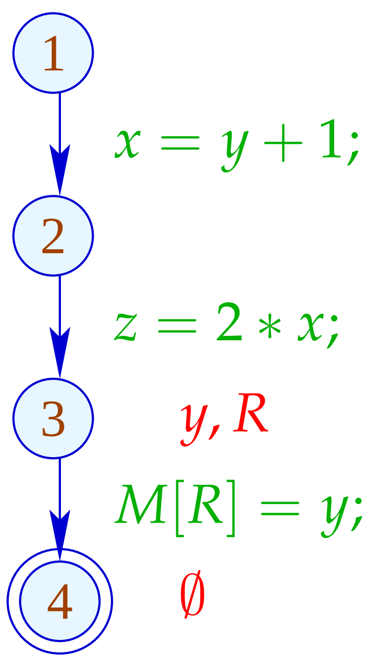
Removal of assignments to dead variables may kill further variables:



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

Warning:

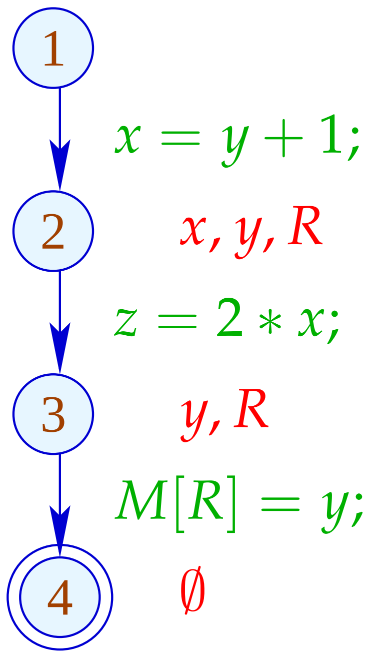
Removal of assignments to dead variables may kill further variables:



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

Warning:

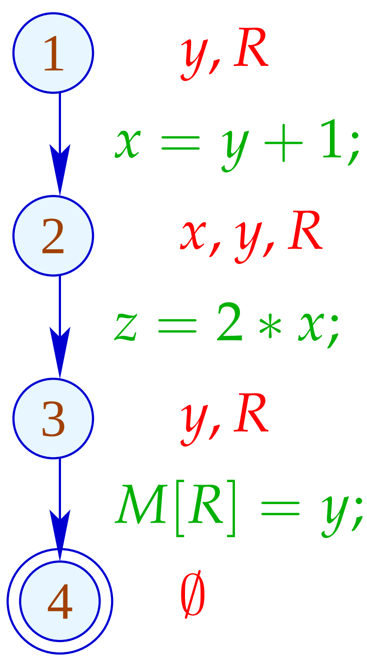
Removal of assignments to dead variables may kill further variables:



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

Warning:

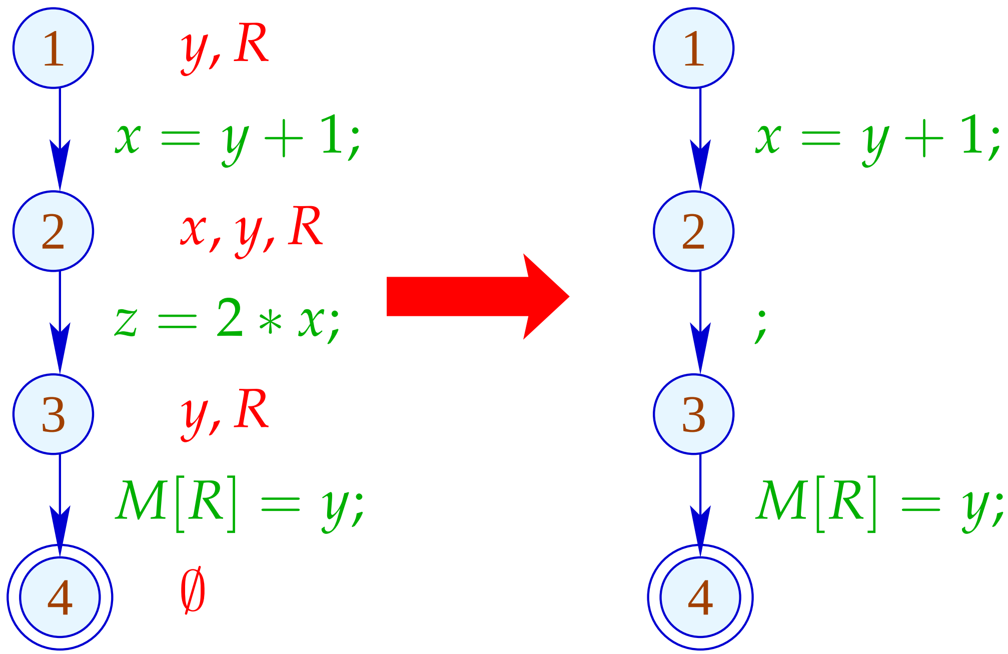
Removal of assignments to dead variables may kill further variables:



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

Warning:

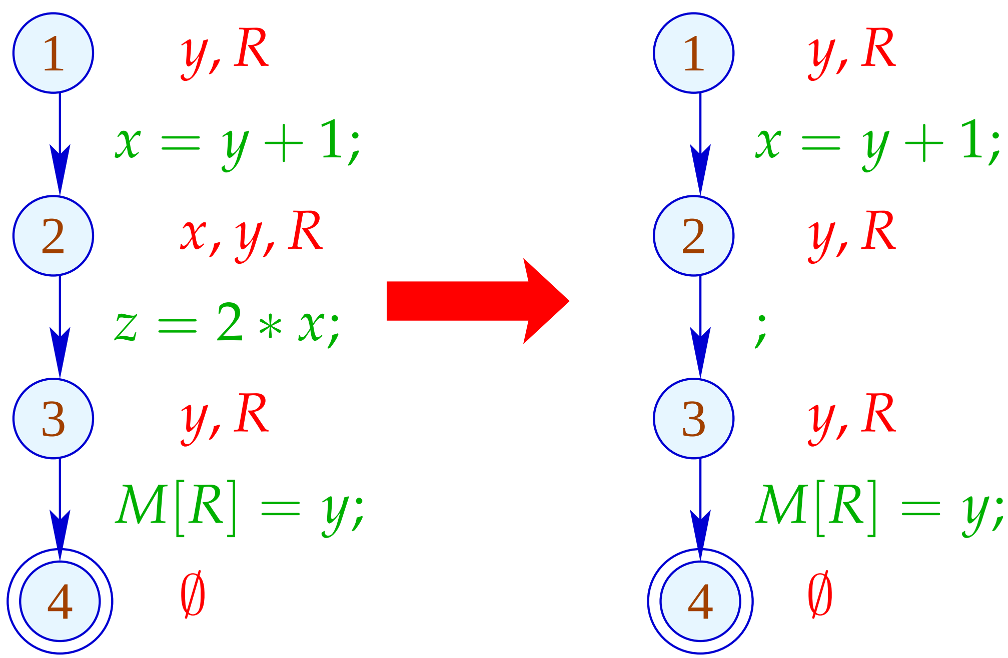
Removal of assignments to dead variables may kill further variables:



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

Warning:

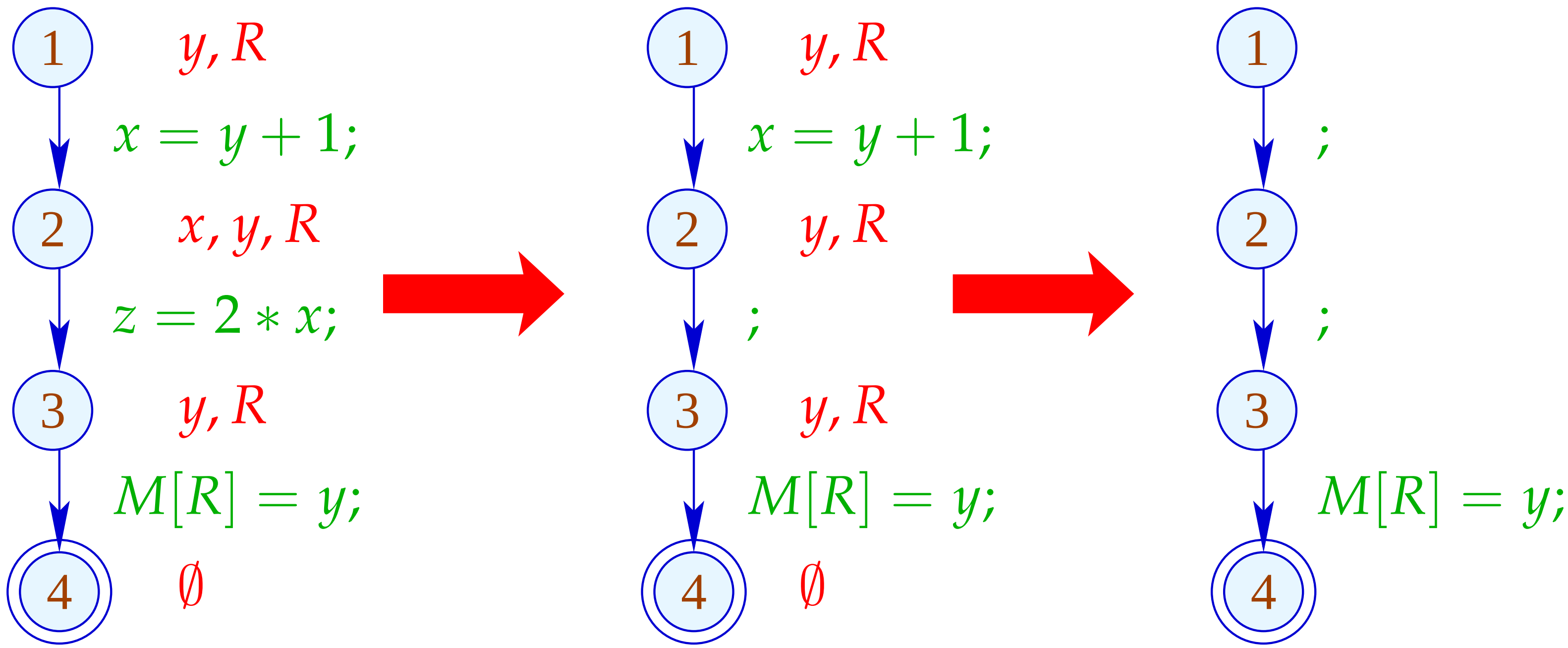
Removal of assignments to dead variables may kill further variables:



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 :-(

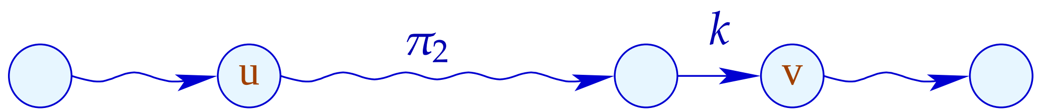
Idea: Analyze true liveness!

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.

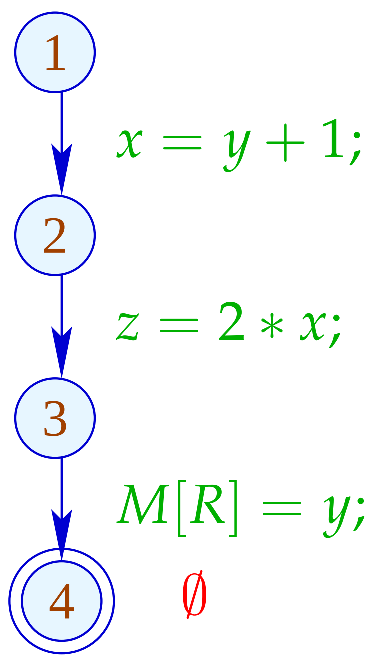


The set of truely used variables at an edge $k = (\_, lab, v)$ is defined as:

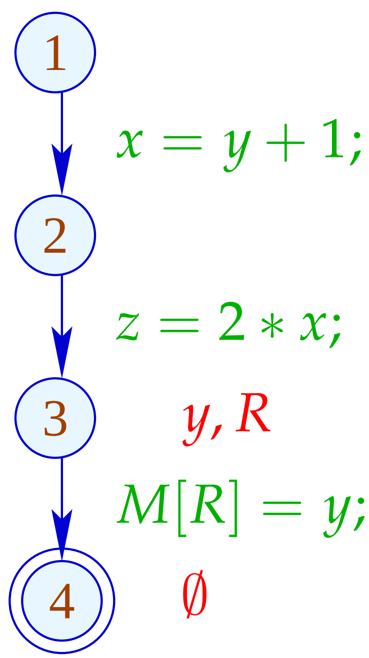
lab	truely used
$;$	$\emptyset$
$\text{Pos}\,(e)$	$Vars\,(e)$
$\text{Neg}\,(e)$	$Vars\,(e)$
$x = e;$	$Vars\,(e)$ $(*)$
$x = M[e];$	$Vars\,(e)$ $(*)$
$M[e_1] = e_2;$	$Vars(e_1) \cup Vars(e_2)$

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

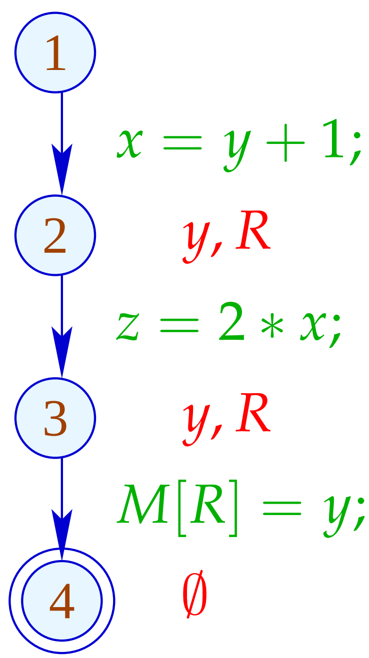
Example:



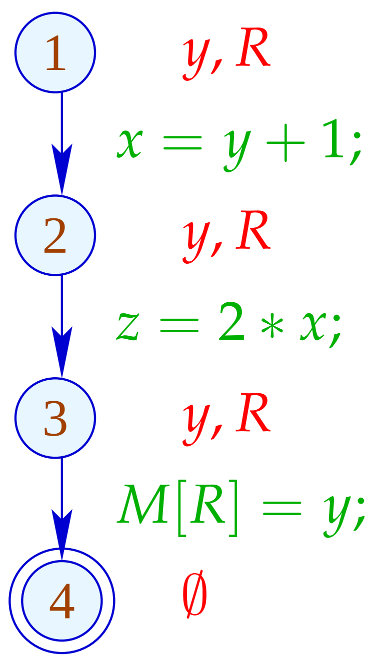
Example:



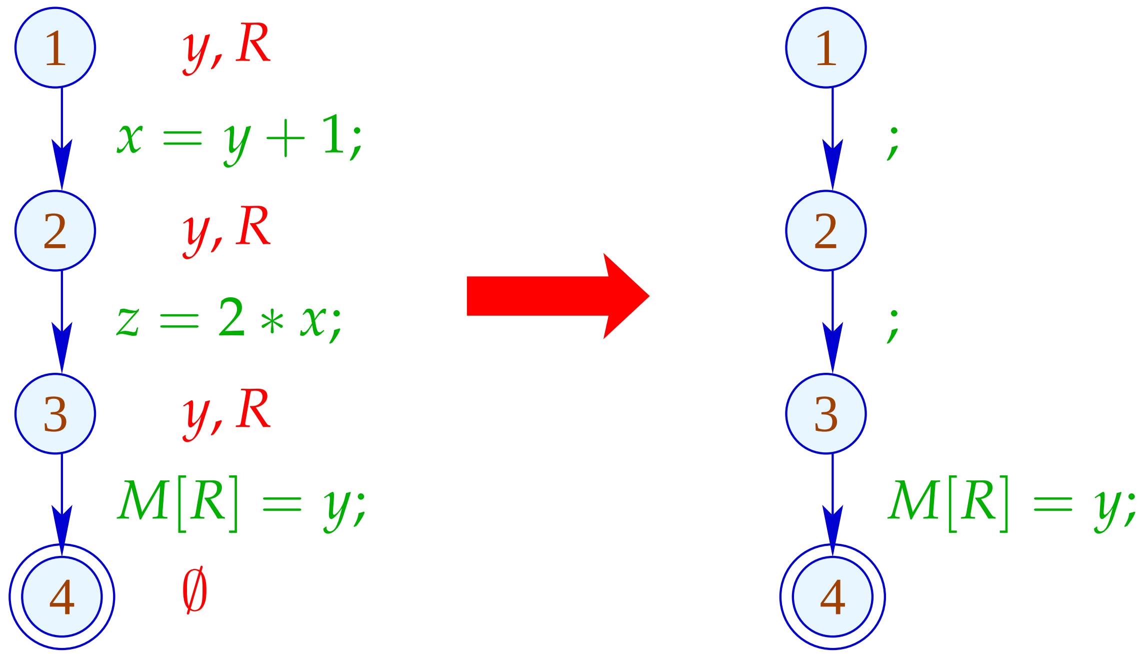
Example:



Example:



Example:



The Effects of Edges:

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

The Effects of Edges:

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

Note:

- The effects of edges for truely live variables are more complicated than for live variables :-)
- Nonetheless, they are distributive !!

Note:

- The effects of edges for truely live variables are more complicated than for live variables :-)
- Nonetheless, they are distributive !!

  To see this, consider for $\mathbb{D} = 2^U$, $f\,y = (u \in y)\,?\,b :\, \emptyset$ We verify:

$$
\begin{array}{lll}
f\,(y_1 \cup y_2) & = & (u \in y_1 \cup y_2)\,?\,b :\, \emptyset \\
 & = & (u \in y_1 \vee u \in y_2)\,?\,b :\, \emptyset \\
 & = & (u \in y_1)\,?\,b :\, \emptyset \cup (u \in y_2)\,?\,b :\, \emptyset \\
 & = & f\,y_1 \cup f\,y_2
\end{array}
$$

Note:

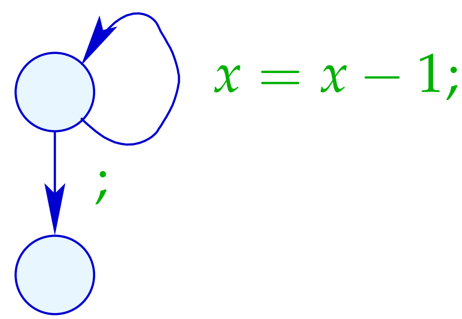
- The effects of edges for truely live variables are more complicated than for live variables :-)
- Nonetheless, they are distributive !!

  To see this, consider for $\mathbb{D} = 2^U$, $f\,y = (u \in y)\,?\,b : \emptyset$ We verify:

$$\begin{aligned} f\,(y_1 \cup y_2) &= (u \in y_1 \cup y_2)\,?\,b : \emptyset \\ &= (u \in y_1 \vee u \in y_2)\,?\,b : \emptyset \\ &= (u \in y_1)\,?\,b : \emptyset \cup (u \in y_2)\,?\,b : \emptyset \\ &= f\,y_1 \cup f\,y_2 \end{aligned}$$

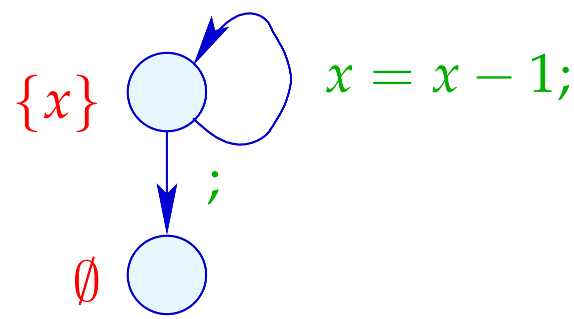
$\Longrightarrow$ the constraint system yields the MOP :-))

- True liveness detects more superfluous assignments than repeated liveness !!!



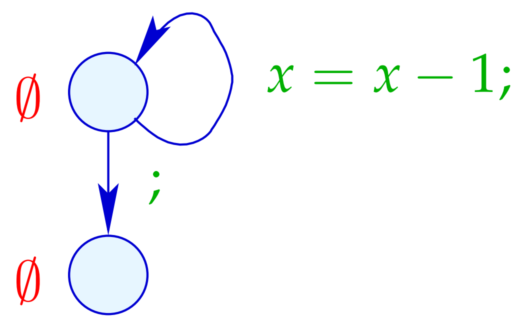
- True liveness detects more superfluous assignments than repeated liveness !!!

Liveness:



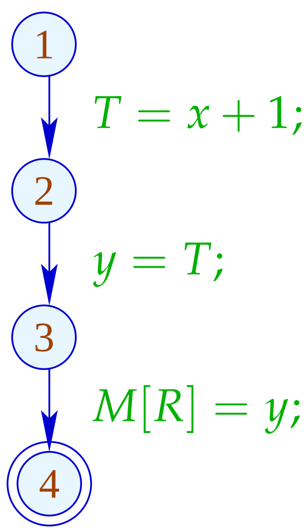
- True liveness detects more superfluous assignments than repeated liveness !!!

True Liveness:



1.3 Removing Superfluous Moves

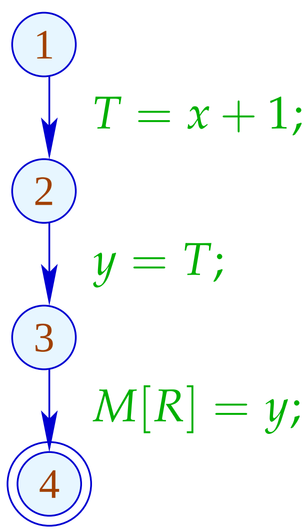
Example:



This variable-variable assignment is obviously useless :-(

1.3 Removing Superfluous Moves

Example:

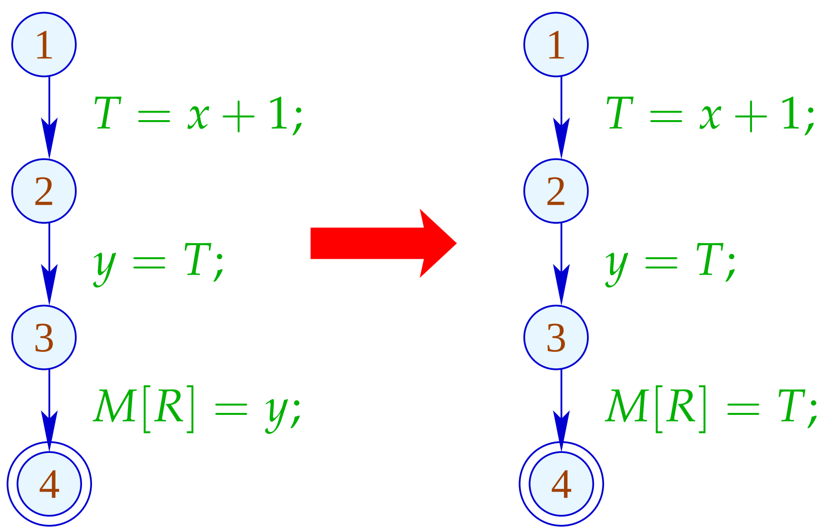


This variable-variable assignment is obviously useless :-(

Instead of y, we could also store T :-)

1.3 Removing Superfluous Moves

Example:

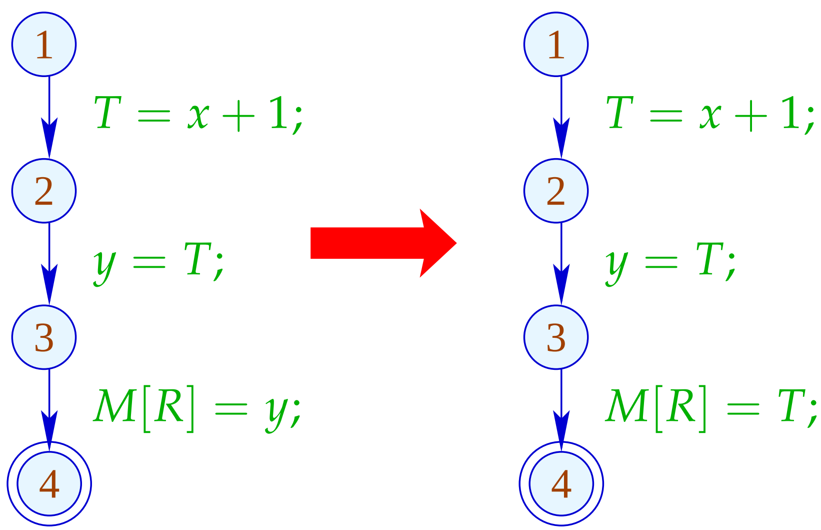


This variable-variable assignment is obviously useless :-(

Instead of y, we could also store T :-)

1.3 Removing Superfluous Moves

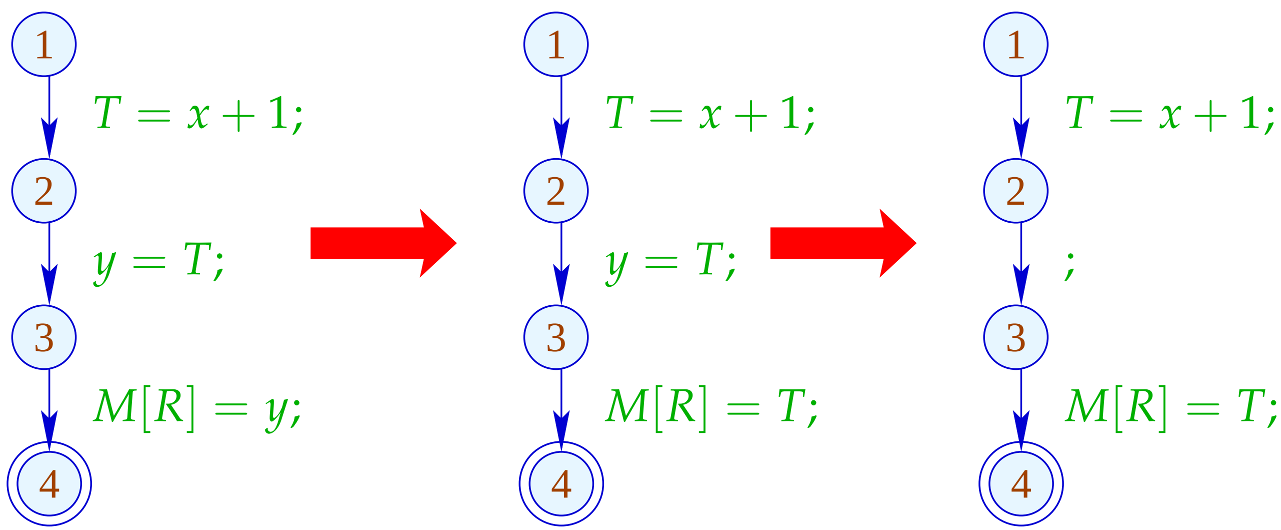
Example:



Advantage: Now, y has become dead :-))

1.3 Removing Superfluous Moves

Example:



Advantage: Now, y has become dead :-))

Idea:

For each expression, we record the variable which currently contains its value :-)

We use: $\mathbb{V} = \mathit{Expr} \to 2^{\mathit{Vars}}$ ...

Idea:

For each expression, we record the variable which currently contains its value :-)

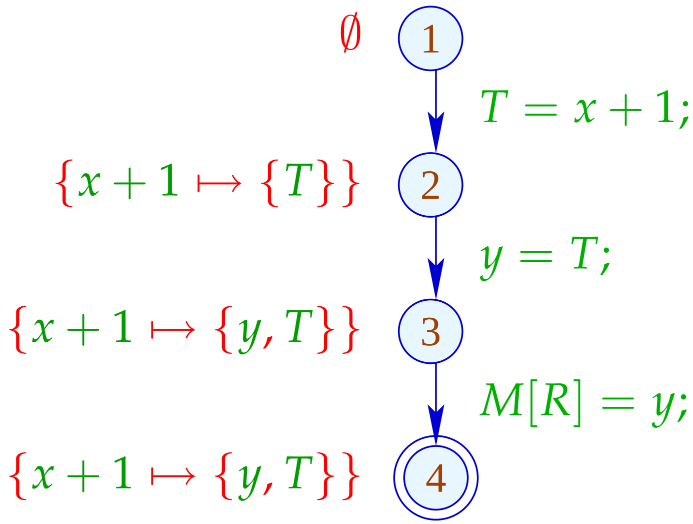
We use: $\mathbb{V} = \mathit{Expr} \to 2^{\mathit{Vars}}$ and define:

$$
\begin{array}{lll}
[\![;]\!]^\sharp\, V & = & V \\
[\![\mathrm{Pos}(e)]\!]^\sharp\, V\, e' & = & [\![\mathrm{Neg}(e)]\!]^\sharp\, V\, e' \quad = \quad \left\{ \begin{array}{ll} \emptyset & \text{if } e' = e \\ V\, e' & \text{otherwise} \end{array} \right. \\
\ldots
\end{array}
$$

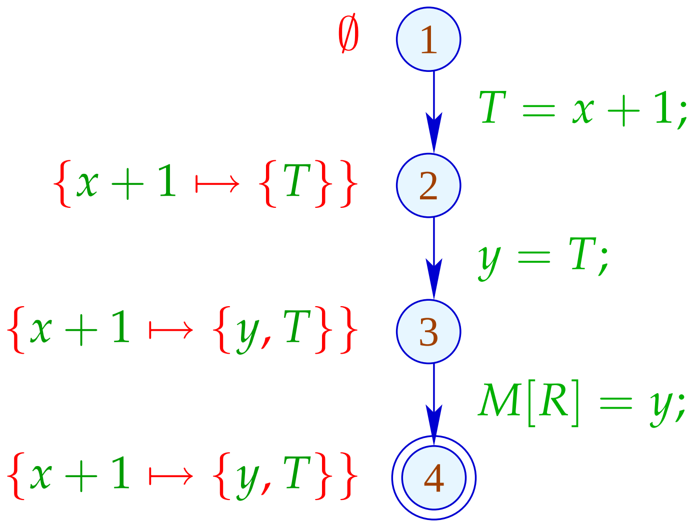
$$
\begin{array}{lcl}
[\![x = c;]\!]^\sharp \, V \, e' & = & \begin{cases} (V\, c) \cup \{x\} & \text{if} \quad e' = c \\ (V\, e') \backslash \{x\} & \text{otherwise} \end{cases} \\
[\![x = y;]\!]^\sharp \, V \, e & = & \begin{cases} (V\, e) \cup \{x\} & \text{if} \quad y \in V\, e \\ (V\, e) \backslash \{x\} & \text{otherwise} \end{cases} \\
[\![x = e;]\!]^\sharp \, V \, e' & = & \begin{cases} \{x\} & \text{if} \quad e' = e \\ (V\, e') \backslash \{x\} & \text{otherwise} \end{cases} \\
[\![x = M[c];]\!]^\sharp \, V \, e' & = & (V\, e') \backslash \{x\} \\
[\![x = M[y];]\!]^\sharp \, V \, e' & = & (V\, e') \backslash \{x\} \\
[\![x = M[e];]\!]^\sharp \, V \, e' & = & \begin{cases} \emptyset & \text{if } e' = e \\ (V\, e') \backslash \{x\} & \text{otherwise} \end{cases}
\end{array}
$$

// analogously for the diverse stores

In the Example:



In the Example:



$\longrightarrow$ We propagate information in forward direction :-)

At *start*, $V_0\, e = \emptyset$ for all e;

$\longrightarrow$ $\sqsubseteq \subseteq \mathbb{V} \times \mathbb{V}$ is defined by:

$$V_1 \sqsubseteq V_2 \quad \text{iff} \quad V_1\, e \supseteq V_2\, e \quad \text{for all } e$$

Observation:

The new effects of edges are distributive:

To show this, we consider the functions:

(1) $f_1^x\, V\, e = (V\, e) \backslash \{x\}$

(2) $f_2^{e,a}\, V = V \oplus \{e \mapsto a\}$

(3) $f_3^{x,y}\, V\, e = (y \in V\, e)\, ?\, (V\, e \cup \{x\}) : ((V\, e) \backslash \{x\})$

Obviously, we have:

$$
\begin{array}{lll}
[\![x = e;]\!]^\sharp & = & f_2^{e,\{x\}} \circ f_1^x \\
[\![x = y;]\!]^\sharp & = & f_3^{x,y} \\
[\![x = M[e];]\!]^\sharp & = & f_2^{e,\emptyset} \circ f_1^x
\end{array}
$$

By closure under composition, the assertion follows :-))

(1) For $f\,V\,e = (V\,e)\backslash\{x\}$, we have:

$$
\begin{array}{lll}
f\,(V_1 \sqcup V_2)\,e & = & ((V_1 \sqcup V_2)\,e)\backslash\{x\} \\
& = & ((V_1\,e) \cap (V_2\,e))\backslash\{x\} \\
& = & ((V_1\,e)\backslash\{x\}) \cap ((V_2\,e)\backslash\{x\}) \\
& = & (f\,V_1\,e) \cap (f\,V_2\,e) \\
& = & (f\,V_1 \sqcup f\,V_2)\,e \qquad \text{:-)}
\end{array}
$$

(2) For $f\,V = V \oplus \{e \mapsto a\}$, we have:

$$
\begin{aligned}
f\,(V_1 \sqcup V_2)\,e' &= ((V_1 \sqcup V_2) \oplus \{e \mapsto a\})\,e' \\
&= (V_1 \sqcup V_2)\,e' \\
&= (f\,V_1 \sqcup f\,V_2)\,e' \qquad \text{given that} \quad e \neq e'
\end{aligned}
$$

$$
\begin{aligned}
f\,(V_1 \sqcup V_2)\,e &= ((V_1 \sqcup V_2) \oplus \{e \mapsto a\})\,e \\
&= a \\
&= ((V_1 \oplus \{e \mapsto a\})\,e) \sqcap ((V_2 \oplus \{e \mapsto a\})\,e) \\
&= (f\,V_1 \sqcup f\,V_2)\,e \qquad \text{:-)}
\end{aligned}
$$

(3) For $f\,V\,e = (y \in V\,e)\,?\,(V\,e \cup \{x\}) : ((V\,e)\backslash\{x\})$, we have:

$$\begin{array}{lll} f\,(V_1 \sqcup V_2)\,e & = & (((V_1 \sqcup V_2)\,e)\backslash\{x\}) \cup (y \in (V_1 \sqcup V_2)\,e)\,?\,\{x\} : \emptyset \\ & = & ((V_1\,e \cap V_2\,e)\backslash\{x\}) \cup (y \in (V_1\,e \cap V_2\,e))\,?\,\{x\} : \emptyset \\ & = & ((V_1\,e \cap V_2\,e)\backslash\{x\}) \cup \\ & & ((y \in V_1\,e)\,?\,\{x\} : \emptyset) \cap ((y \in V_2\,e)\,?\,\{x\} : \emptyset) \\ & = & (((V_1\,e)\backslash\{x\}) \cup (y \in V_1\,e)\,?\,\{x\} : \emptyset) \cap \\ & & (((V_2\,e)\backslash\{x\}) \cup (y \in V_2\,e)\,?\,\{x\} : \emptyset) \\ & = & (f\,V_1 \sqcup f\,V_2)\,e \qquad\qquad :\text{-)} \end{array}$$