

## 1.2 Removing Assignments to Dead Variables

Example:

1 :  $x = y + 2;$

2 :  $y = 5;$

3 :  $x = y + 3;$

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:

- Assignments to dead variables can be removed
- Such inefficiencies may originate from other transformations.

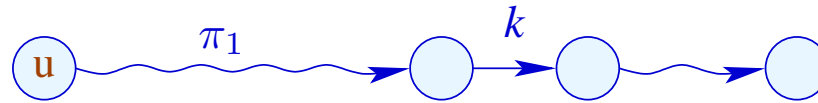
## Formal Definition:

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

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

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

- $k$  is a **use** of  $x$ ; and
- $\pi_1$  does not contain a **definition** of  $x$ .

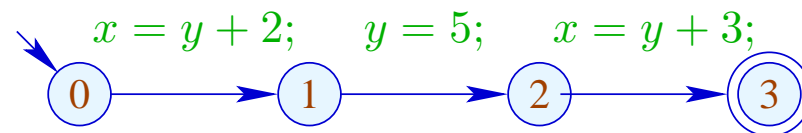


Thereby, the set of all defined or used variables at an edge  $k = (\_, lab, \_)$  is defined by:

<i>lab</i>	used	defined
;	$\emptyset$	$\emptyset$
$Pos(e)$	$Vars(e)$	$\emptyset$
$Neg(e)$	$Vars(e)$	$\emptyset$
$x = e;$	$Vars(e)$	$\{x\}$
$x = M[e];$	$Vars(e)$	$\{x\}$
$M[e_1] = e_2;$	$Vars(e_1) \cup Vars(e_2)$	$\emptyset$

A variable  $x$  which is not live at  $u$  along  $\pi$  (relative to  $X$ ) is called **dead** at  $u$  along  $\pi$  (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  $[[k]]^\#$  which transforms the set of variables which are live at  $v$  into the set of variables which are live at  $u$  ...

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

For  $k = (\_, lab, \_)$  , define  $\llbracket k \rrbracket^\# = \llbracket lab \rrbracket^\#$  by:

$$\begin{aligned}\llbracket ; \rrbracket^\# L &= L \\ \llbracket \text{Pos}(e) \rrbracket^\# L &= \llbracket \text{Neg}(e) \rrbracket^\# L = L \cup Vars(e) \\ \llbracket x = e; \rrbracket^\# L &= (L \setminus \{x\}) \cup Vars(e) \\ \llbracket x = M[e]; \rrbracket^\# L &= (L \setminus \{x\}) \cup Vars(e) \\ \llbracket M[e_1] = e_2; \rrbracket^\# L &= L \cup Vars(e_1) \cup Vars(e_2)\end{aligned}$$

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

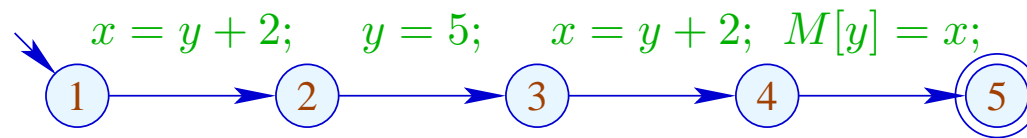
For  $k = (\_, lab, \_)$  , define  $\llbracket k \rrbracket^\# = \llbracket lab \rrbracket^\#$  by:

$$\begin{aligned}\llbracket ; \rrbracket^\# L &= L \\ \llbracket Pos(e) \rrbracket^\# L &= \llbracket Neg(e) \rrbracket^\# L = L \cup Vars(e) \\ \llbracket x = e; \rrbracket^\# L &= (L \setminus \{x\}) \cup Vars(e) \\ \llbracket x = M[e]; \rrbracket^\# L &= (L \setminus \{x\}) \cup Vars(e) \\ \llbracket M[e_1] = e_2; \rrbracket^\# L &= L \cup Vars(e_1) \cup Vars(e_2)\end{aligned}$$

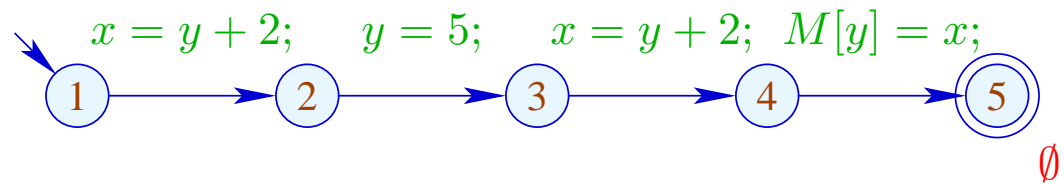
$\llbracket k \rrbracket^\#$  can again be composed to the effects of  $\llbracket \pi \rrbracket^\#$  of paths  $\pi = k_1 \dots k_r$  by:

$$\llbracket \pi \rrbracket^\# = \llbracket k_1 \rrbracket^\# \circ \dots \circ \llbracket k_r \rrbracket^\#$$

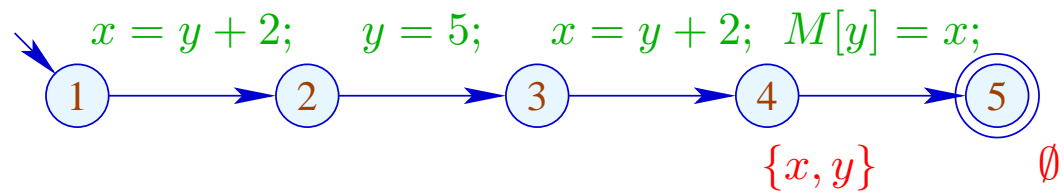
We verify that these definitions are **meaningful**



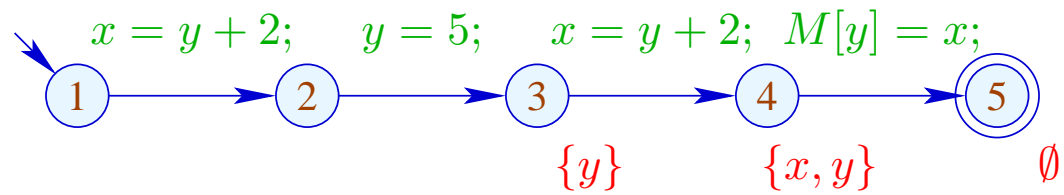
We verify that these definitions are **meaningful**



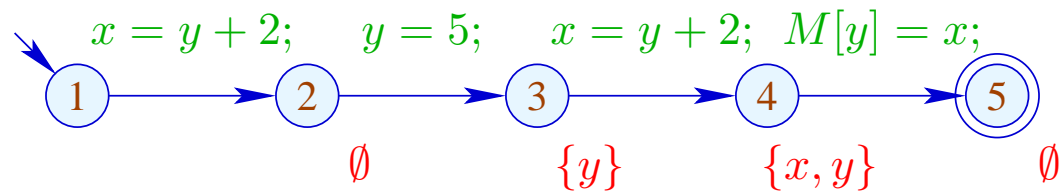
We verify that these definitions are **meaningful**



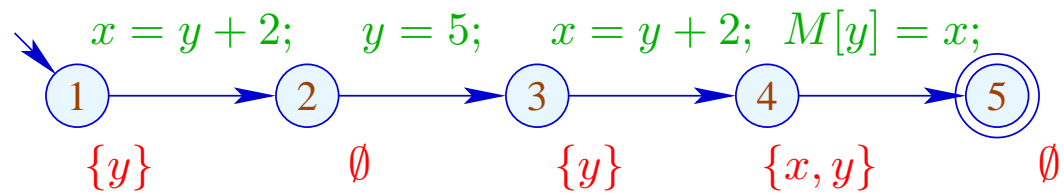
We verify that these definitions are **meaningful**



We verify that these definitions are **meaningful**



We verify that these definitions are **meaningful**



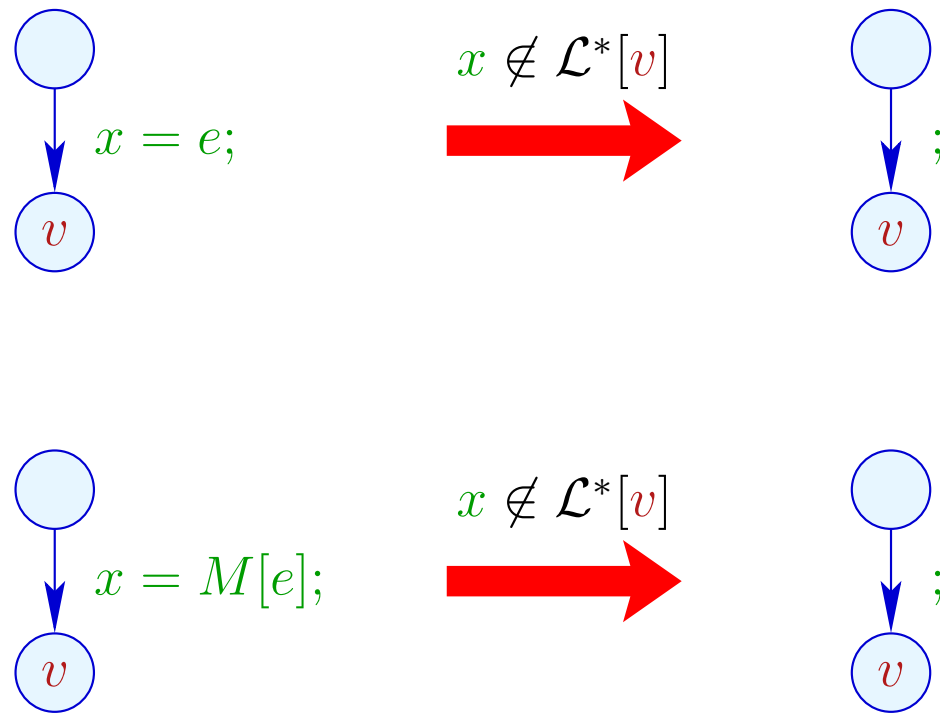
The set of variables which are live at  $u$  then is given by:

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

... literally:

- The paths **start** in  $u$   
 $\implies$  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 DE (Dead assignment Elimination):



## Correctness Proof:

- **Correctness of the effects of edges:** If  $L$  is the set of variables which are live at the exit of the path  $\pi$ , then  $[[\pi]]^\# L$  is the set of variables which are live at the beginning of  $\pi$
- **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:

$$\begin{aligned}\mathcal{L}[\textit{stop}] &\supseteq X \\ \mathcal{L}[u] &\supseteq \llbracket k \rrbracket^\# (\mathcal{L}[v]) \quad k = (u, \_, v) \text{ edge}\end{aligned}$$

(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  $\llbracket k \rrbracket^\#$  are distributive

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

(1) Collecting constraints:

$$\begin{aligned}\mathcal{L}[\textit{stop}] &\supseteq X \\ \mathcal{L}[u] &\supseteq \llbracket k \rrbracket^\# (\mathcal{L}[v]) \quad k = (u, \_, v) \text{ edge}\end{aligned}$$

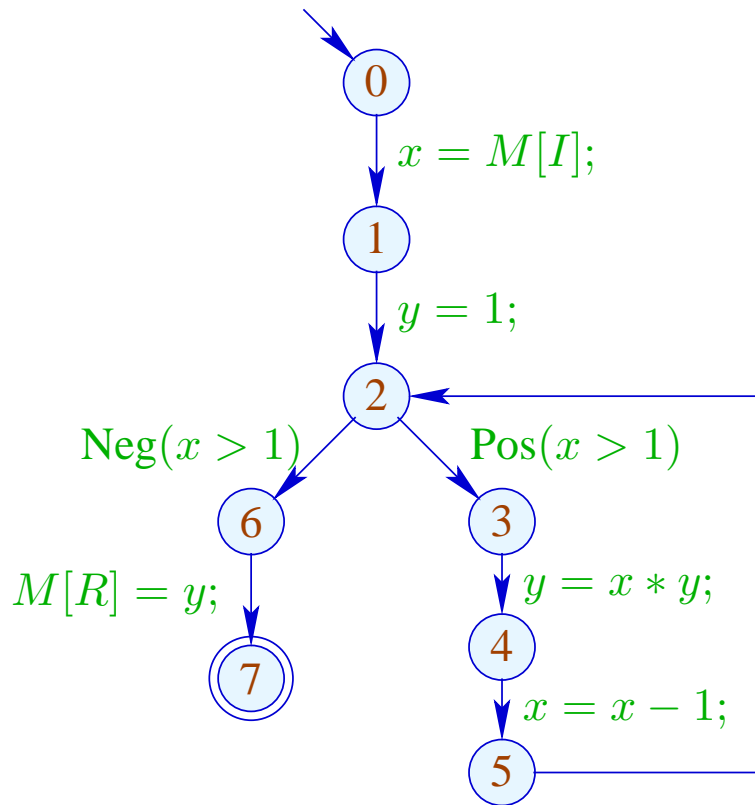
(2) Solving the constraint system by means of RR iteration.

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  $\llbracket k \rrbracket^\#$  are distributive

**Caveat:** The information is propagated **backwards** !!!

## Example:



$$\mathcal{L}[0] \supseteq (\mathcal{L}[1] \setminus \{x\}) \cup \{I\}$$

$$\mathcal{L}[1] \supseteq \mathcal{L}[2] \setminus \{y\}$$

$$\mathcal{L}[2] \supseteq (\mathcal{L}[6] \cup \{x\}) \cup (\mathcal{L}[3] \cup \{x\})$$

$$\mathcal{L}[3] \supseteq (\mathcal{L}[4] \setminus \{y\}) \cup \{x, y\}$$

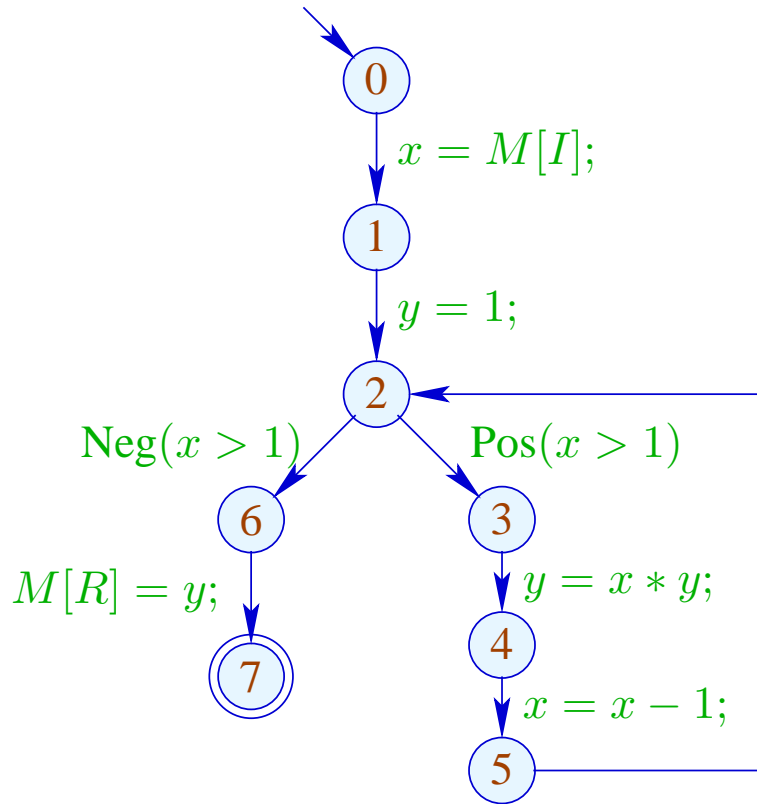
$$\mathcal{L}[4] \supseteq (\mathcal{L}[5] \setminus \{x\}) \cup \{x\}$$

$$\mathcal{L}[5] \supseteq \mathcal{L}[2]$$

$$\mathcal{L}[6] \supseteq \mathcal{L}[7] \cup \{y, R\}$$

$$\mathcal{L}[7] \supseteq \emptyset$$

## Example:

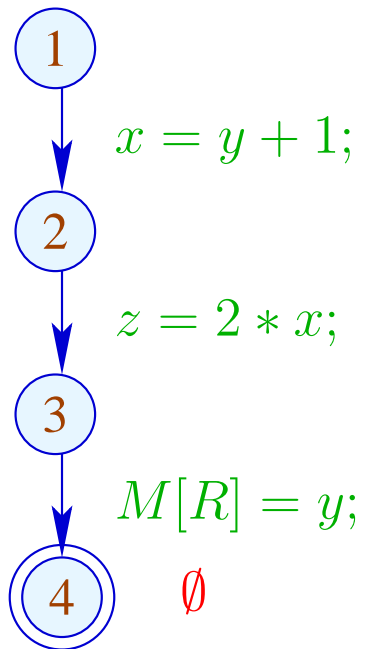


	1	2
7	$\emptyset$	
6	$\{y, R\}$	
2	$\{x, y, R\}$	dito
5	$\{x, y, R\}$	
4	$\{x, y, R\}$	
3	$\{x, y, R\}$	
1	$\{x, R\}$	
0	$\{I, R\}$	

The left-hand side of no assignment is **dead**

**Caveat:**

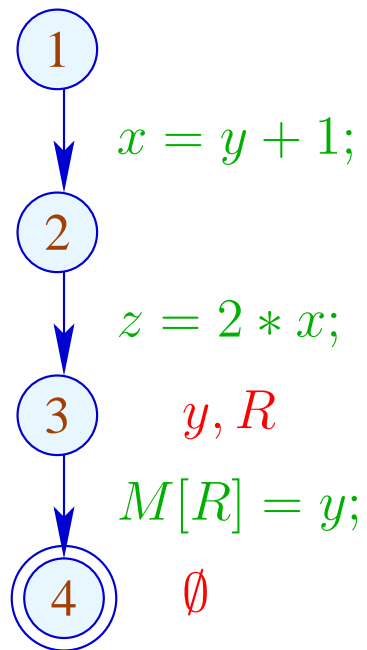
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is **dead**

**Caveat:**

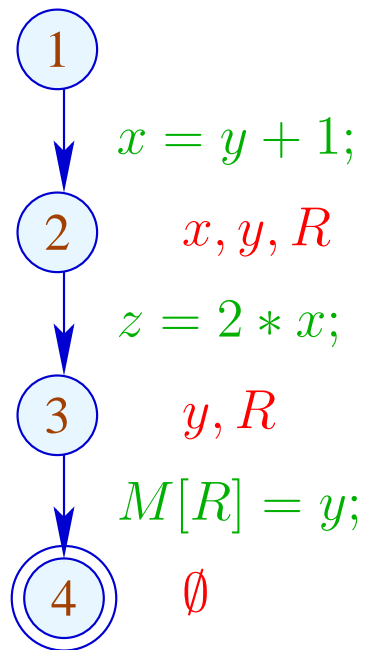
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is **dead**

### Caveat:

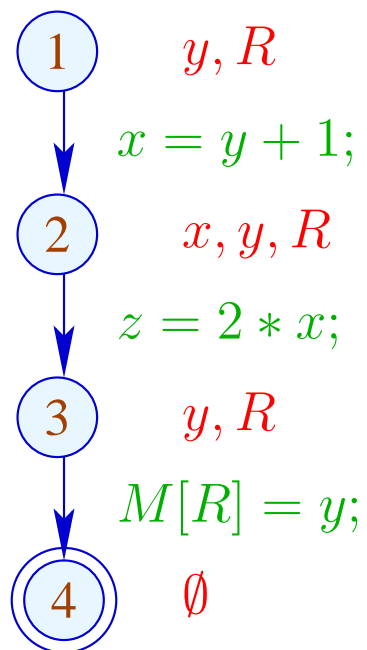
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is **dead**

### Caveat:

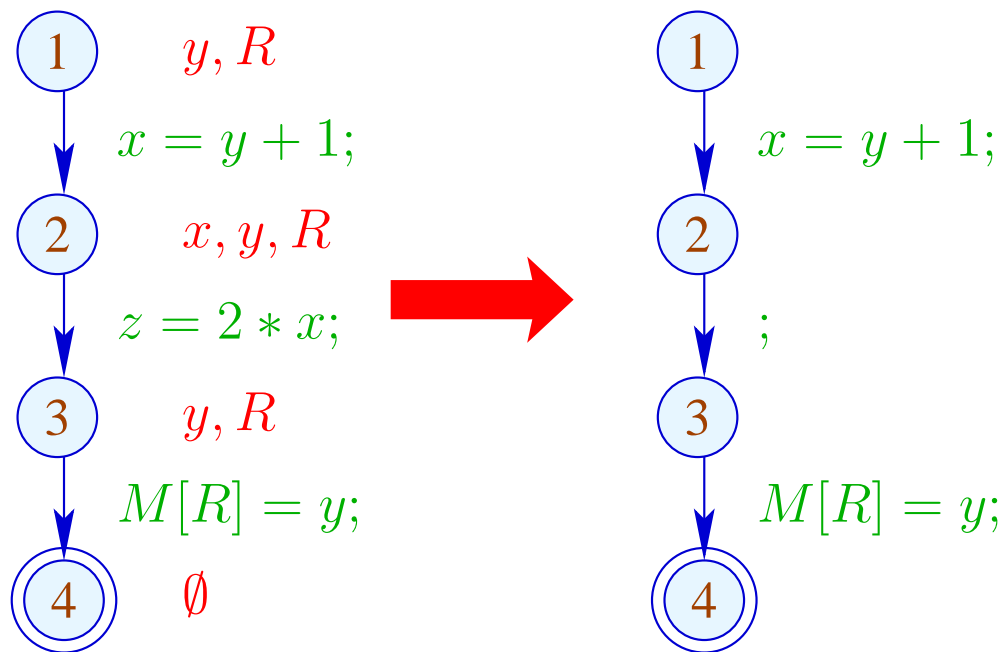
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is **dead**

**Caveat:**

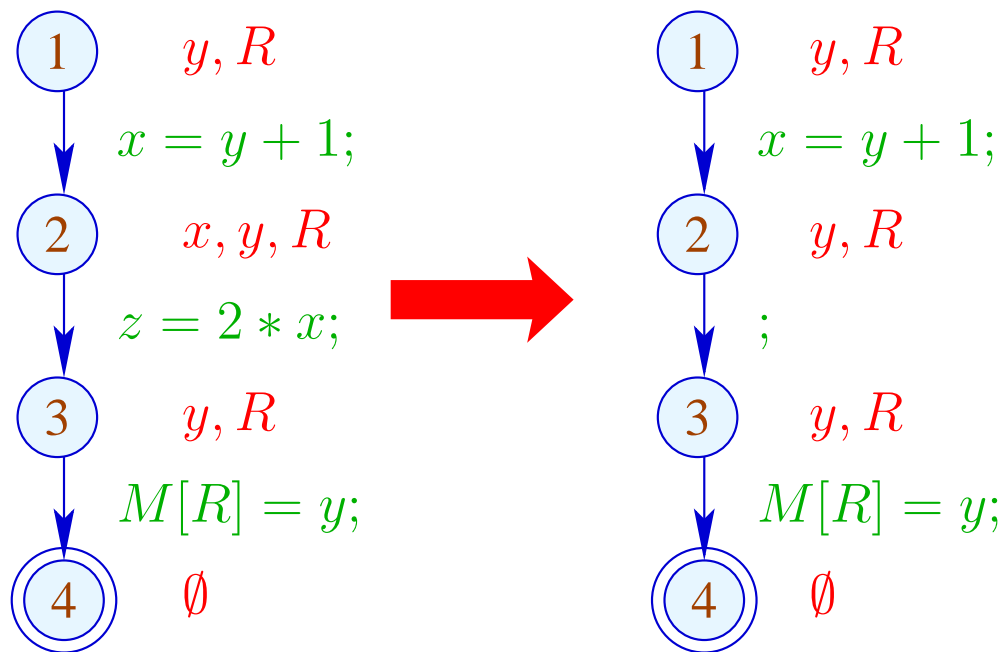
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is **dead**

**Caveat:**

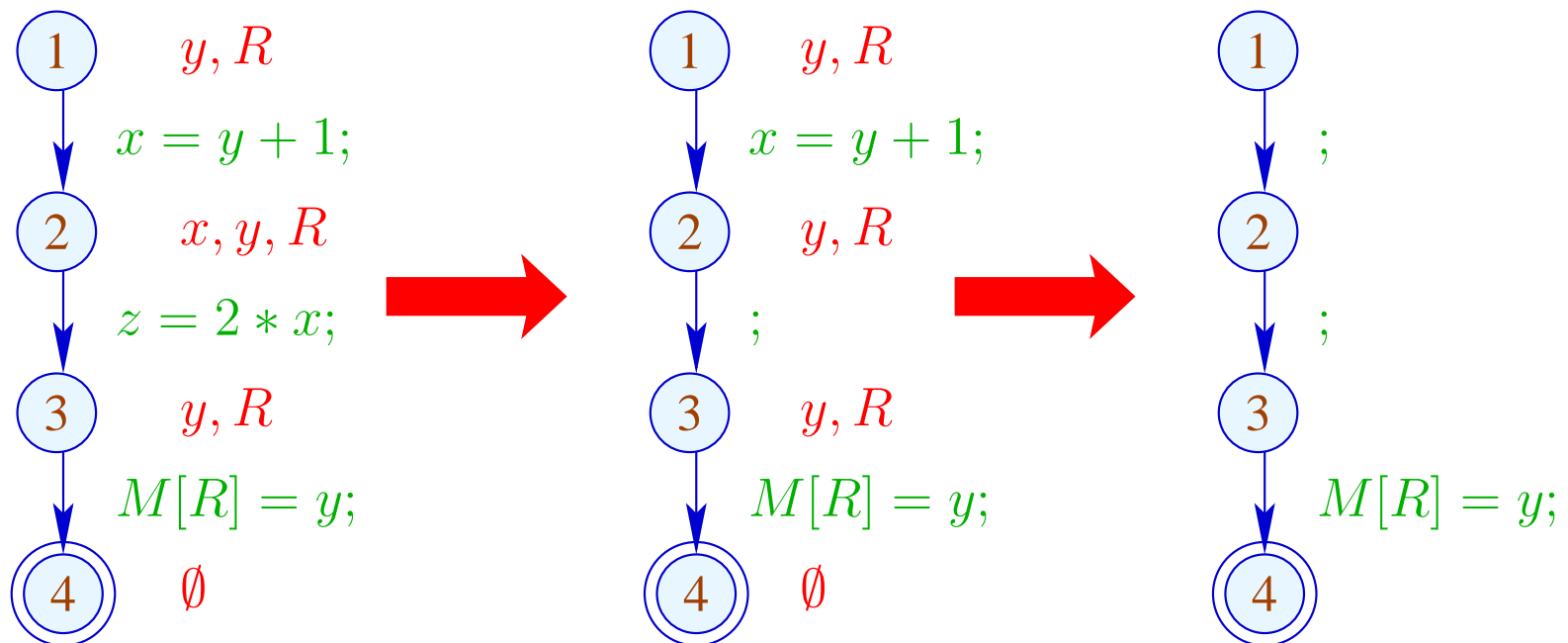
Removal of assignments to dead variables may kill further variables:



The left-hand side of no assignment is **dead**

**Caveat:**

Removal of assignments to dead variables may kill further variables:



Re-analyzing the program is inconvenient

**Idea:** Analyze **true** liveness!

$x$  is called **truly live** at  $u$  along a path  $\pi$  (relative to  $X$ ), either

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

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

- $k$  is a **true** use of  $x$ ;
- $\pi_1$  does not contain any **definition** of  $x$ .

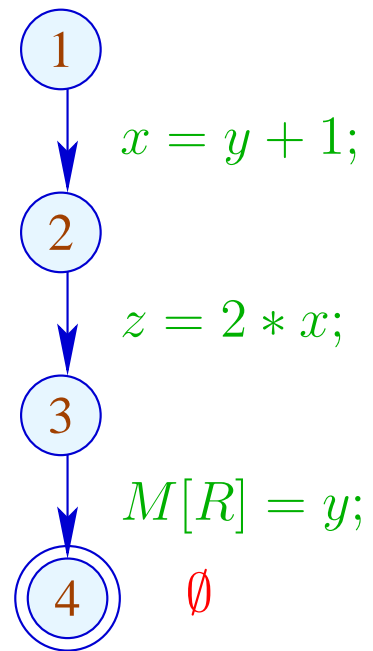


The set of truly used variables at an edge  $k = (\_, lab, v)$  is defined as:

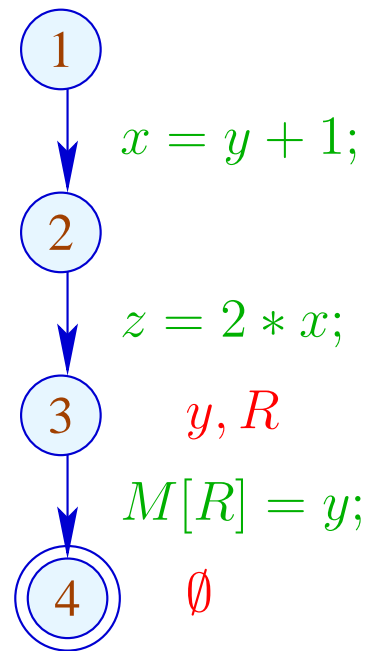
<i>lab</i>	truly used
;	$\emptyset$
$Pos(e)$	$Vars(e)$
$Neg(e)$	$Vars(e)$
$x = e;$	$Vars(e)$ (*)
$x = M[e];$	$Vars(e)$ (*)
$M[e_1] = e_2;$	$Vars(e_1) \cup Vars(e_2)$

(\*) – given that  $x$  is truly live at  $v$

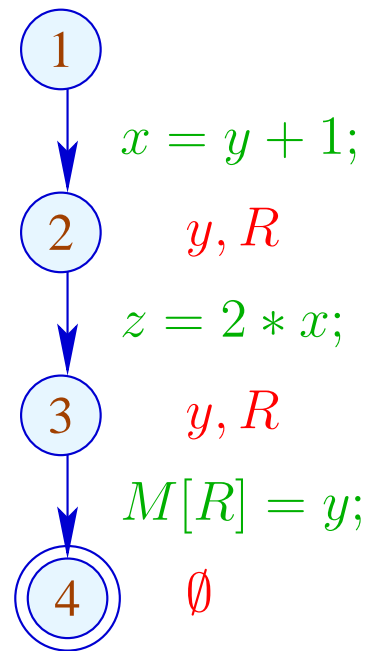
Example:



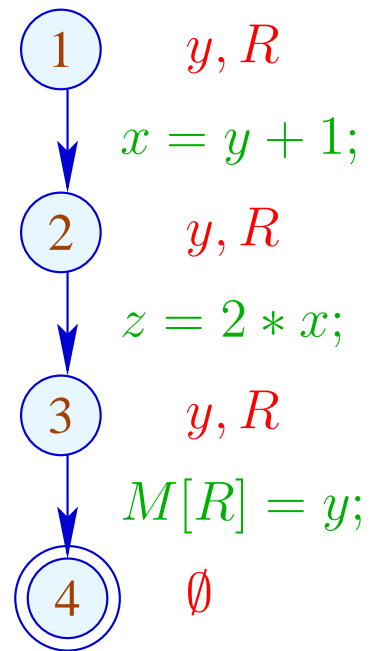
Example:



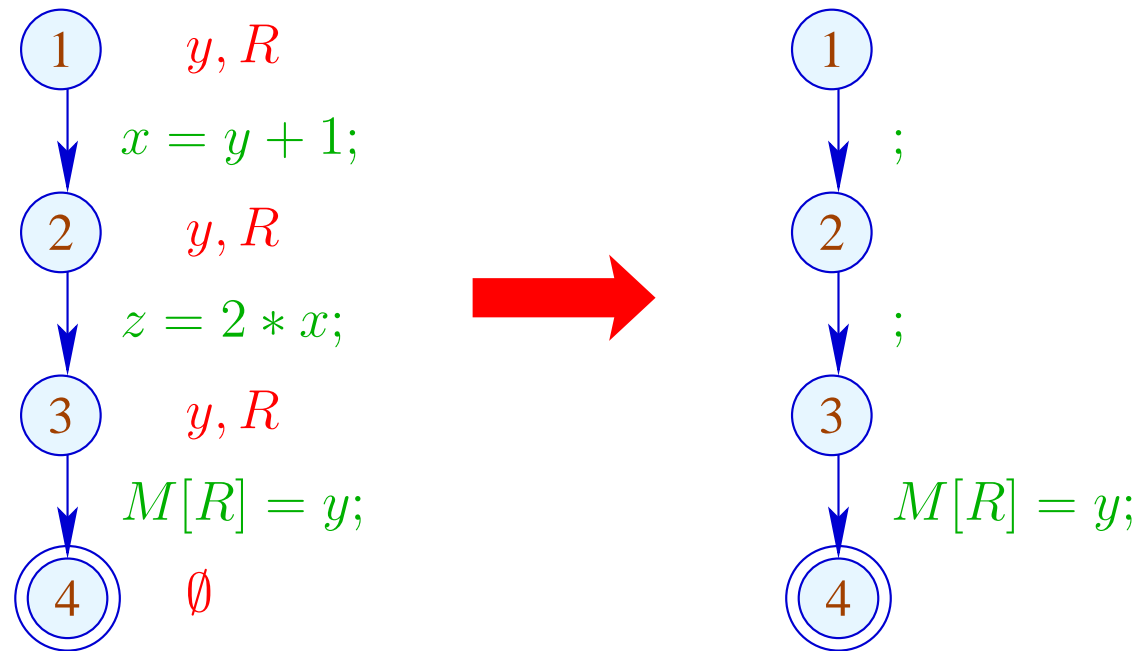
Example:



Example:



Example:



## The Effects of Edges:

$$\begin{aligned} \llbracket ; \rrbracket^\# L &= L \\ \llbracket \text{Pos}(e) \rrbracket^\# L &= \llbracket \text{Neg}(e) \rrbracket^\# L = L \cup \text{Vars}(e) \\ \llbracket x = e; \rrbracket^\# L &= (L \setminus \{x\}) \cup \text{Vars}(e) \\ \llbracket x = M[e]; \rrbracket^\# L &= (L \setminus \{x\}) \cup \text{Vars}(e) \\ \llbracket M[e_1] = e_2; \rrbracket^\# L &= L \cup \text{Vars}(e_1) \cup \text{Vars}(e_2) \end{aligned}$$

## The Effects of Edges:

$$\begin{aligned} \llbracket ; \rrbracket^\# L &= L \\ \llbracket \text{Pos}(e) \rrbracket^\# L &= \llbracket \text{Neg}(e) \rrbracket^\# L = L \cup \text{Vars}(e) \\ \llbracket x = e; \rrbracket^\# L &= (L \setminus \{x\}) \cup (x \in L) ? \text{Vars}(e) : \emptyset \\ \llbracket x = M[e]; \rrbracket^\# L &= (L \setminus \{x\}) \cup (x \in L) ? \text{Vars}(e) : \emptyset \\ \llbracket M[e_1] = e_2; \rrbracket^\# L &= L \cup \text{Vars}(e_1) \cup \text{Vars}(e_2) \end{aligned}$$

## Note:

- The effects of edges for truly live variables are **more complicated** than for live variables
- Nonetheless, they are **distributive !!**

## Note:

- The effects of edges for truly 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:

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

## Note:

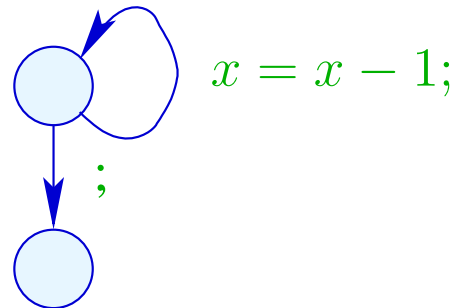
- The effects of edges for truly live variables are **more complicated** than for live variables
- Nonetheless, they are **distributive !!**

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

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

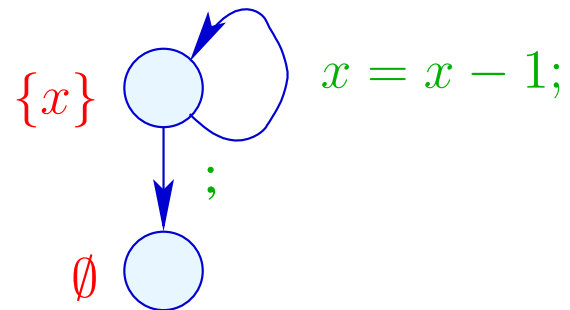
$\implies$  the constraint system yields the **MOP**

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



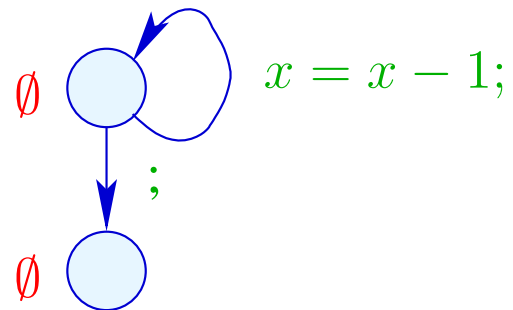
- True liveness detects **more** superfluous assignments than repeated liveness !!!

Liveness:



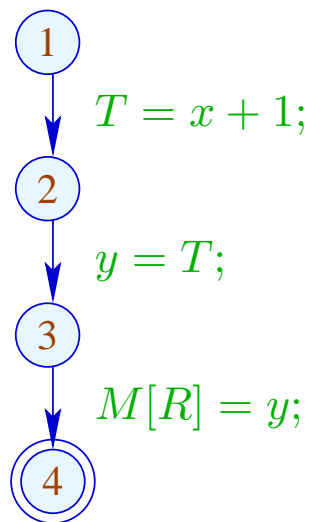
- True liveness detects **more** superfluous assignments than repeated liveness !!!

True Liveness:



## 1.3 Removing Superfluous Moves

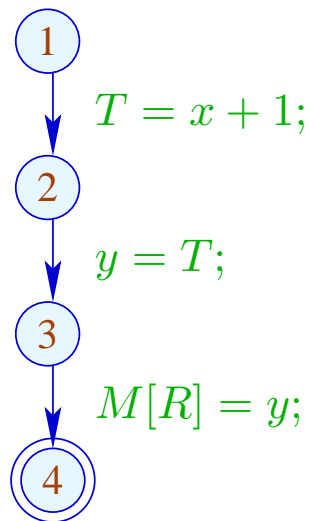
Example:



This variable-variable assignment is obviously useless

## 1.3 Removing Superfluous Moves

Example:

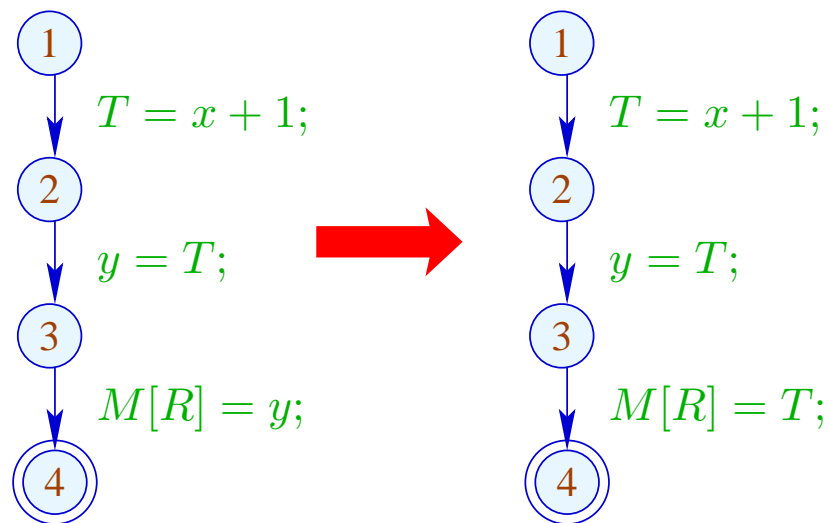


This variable-variable assignment is obviously useless

Instead of  $y$ , we could also store  $T$

## 1.3 Removing Superfluous Moves

Example:

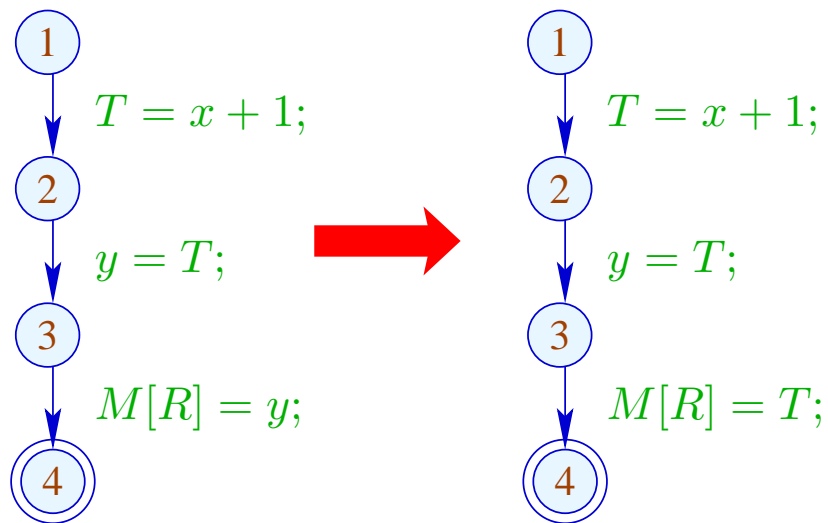


This variable-variable assignment is obviously useless

Instead of  $y$ , we could also store  $T$

## 1.3 Removing Superfluous Moves

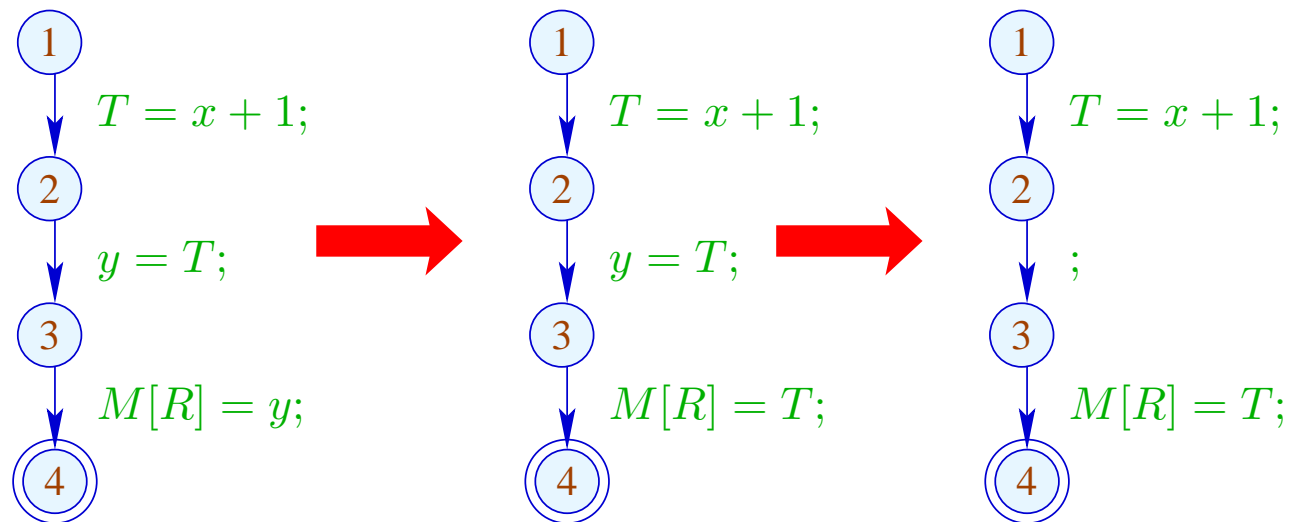
Example:



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:  $\mathbb{V} = \text{Expr} \rightarrow 2^{\text{Vars}} \dots$

## Idea:

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

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

$$\llbracket ; \rrbracket^{\#} V = V$$

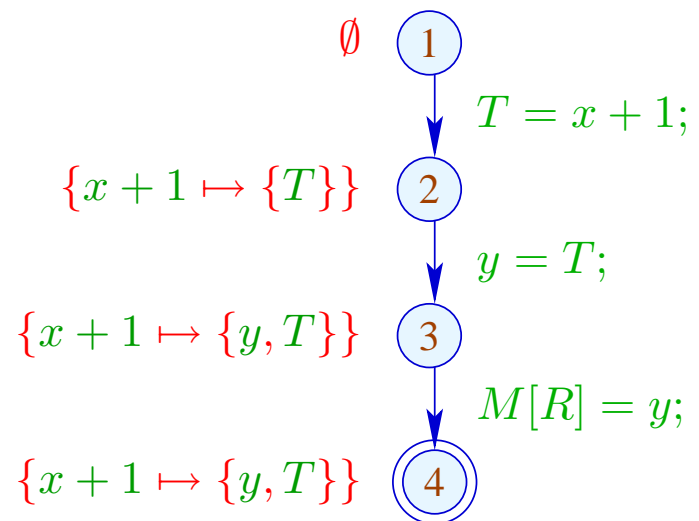
$$\llbracket \text{Pos}(e) \rrbracket^{\#} V e' = \llbracket \text{Neg}(e) \rrbracket^{\#} V e' = \begin{cases} \emptyset & \text{if } e' = e \\ V e' & \text{otherwise} \end{cases}$$

...

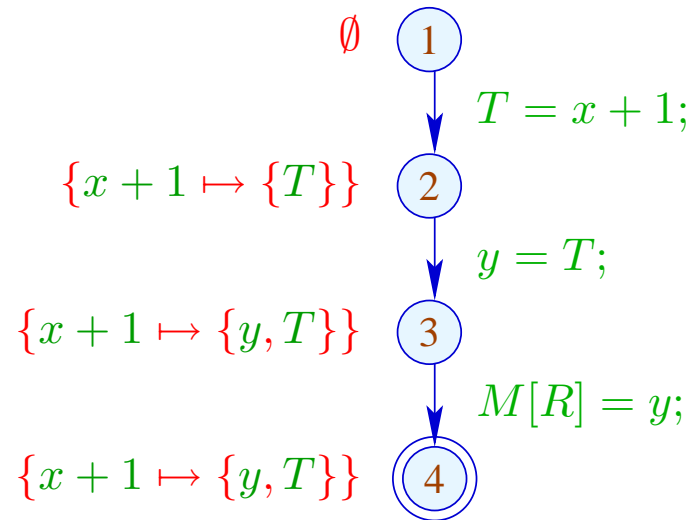
$$\begin{aligned}
[[x = c;]]^\# V e' &= \begin{cases} (V c) \cup \{x\} & \text{if } e' = c \\ (V e') \setminus \{x\} & \text{otherwise} \end{cases} \\
[[x = y;]]^\# V e &= \begin{cases} (V e) \cup \{x\} & \text{if } y \in V e \\ (V e) \setminus \{x\} & \text{otherwise} \end{cases} \\
[[x = e;]]^\# V e' &= \begin{cases} \{x\} & \text{if } e' = e \\ (V e') \setminus \{x\} & \text{otherwise} \end{cases} \\
[[x = M[c];]]^\# V e' &= (V e') \setminus \{x\} \\
[[x = M[y];]]^\# V e' &= (V e') \setminus \{x\} \\
[[x = M[e];]]^\# V e' &= \begin{cases} \emptyset & \text{if } e' = e \\ (V e') \setminus \{x\} & \text{otherwise} \end{cases}
\end{aligned}$$

// analogously for the diverse stores

In the Example:



## In the Example:



→ We propagate information in **forward** direction

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

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

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

## Observation:

The new effects of edges are **distributive**:

To show this, we consider the functions:

- (1)  $f_1^x V e = (V e) \setminus \{x\}$
- (2)  $f_2^{e,a} V = V \oplus \{e \mapsto a\}$
- (3)  $f_3^{x,y} V e = (y \in V e) ? (V e \cup \{x\}) : ((V e) \setminus \{x\})$

Obviously, we have:

$$\begin{aligned} \llbracket x = e; \rrbracket^\# &= f_2^{e, \{x\}} \circ f_1^x \\ \llbracket x = y; \rrbracket^\# &= f_3^{x,y} \\ \llbracket x = M[e]; \rrbracket^\# &= f_2^{e, \emptyset} \circ f_1^x \end{aligned}$$

By closure under **composition**, the assertion follows

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

$$\begin{aligned} f (V_1 \sqcup V_2) e &= ((V_1 \sqcup V_2) e) \setminus \{x\} \\ &= ((V_1 e) \cap (V_2 e)) \setminus \{x\} \\ &= ((V_1 e) \setminus \{x\}) \cap ((V_2 e) \setminus \{x\}) \\ &= (f V_1 e) \cap (f V_2 e) \\ &= (f V_1 \sqcup f V_2) e \end{aligned}$$

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

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

$$\begin{aligned}
 f(V_1 \sqcup V_2) e &= ((V_1 \sqcup V_2) \oplus \{e \mapsto a\}) e \\
 &= a \\
 &= ((V_1 \oplus \{e \mapsto a\}) e) \cap ((V_2 \oplus \{e \mapsto a\}) e) \\
 &= (f V_1 \sqcup f V_2) e
 \end{aligned}$$

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

$$\begin{aligned}
 f (V_1 \sqcup V_2) e &= (((V_1 \sqcup V_2) e) \setminus \{x\}) \cup (y \in (V_1 \sqcup V_2) e) ? \{x\} : \emptyset \\
 &= ((V_1 e \cap V_2 e) \setminus \{x\}) \cup (y \in (V_1 e \cap V_2 e)) ? \{x\} : \emptyset \\
 &= ((V_1 e \cap V_2 e) \setminus \{x\}) \cup \\
 &\quad ((y \in V_1 e) ? \{x\} : \emptyset) \cap ((y \in V_2 e) ? \{x\} : \emptyset) \\
 &= (((V_1 e) \setminus \{x\}) \cup (y \in V_1 e) ? \{x\} : \emptyset) \cap \\
 &\quad (((V_2 e) \setminus \{x\}) \cup (y \in V_2 e) ? \{x\} : \emptyset) \\
 &= (f V_1 \sqcup f V_2) e
 \end{aligned}$$

## We conclude:

→ Solving the constraint system returns the MOP solution

→ Let  $\mathcal{V}$  denote this solution.

If  $x \in \mathcal{V}[u]e$ , then  $x$  at  $u$  contains the value of  $e$  —  
which we have stored in  $T_e$

⇒

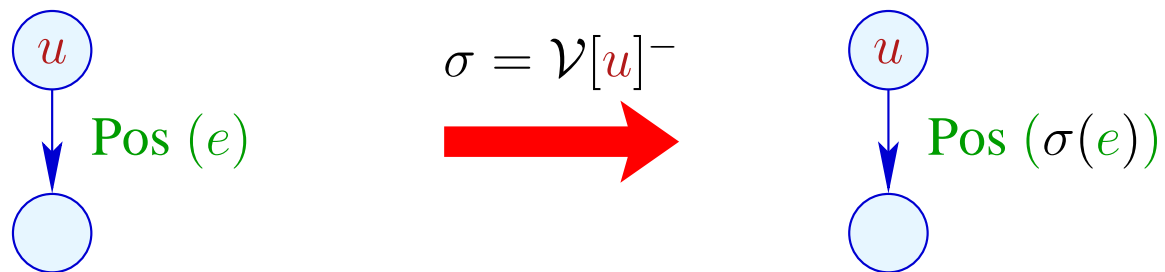
the access to  $x$  can be replaced by the access to  $T_e$

For  $V \in \mathbb{V}$ , let  $V^-$  denote the **variable substitution** with:

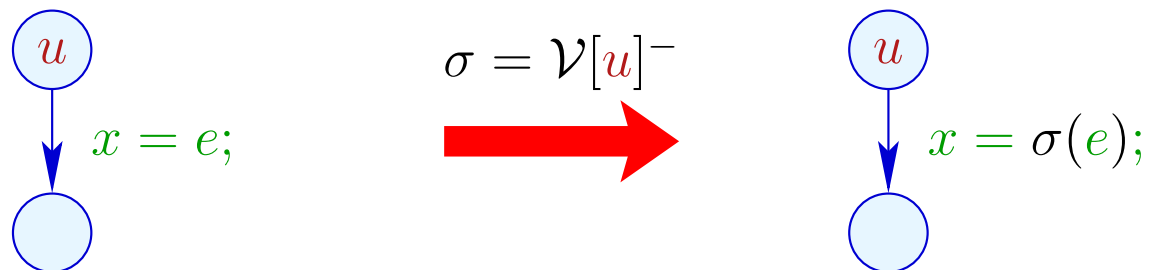
$$V^- x = \begin{cases} T_e & \text{if } x \in V e \\ x & \text{otherwise} \end{cases}$$

if  $V e \cap V e' = \emptyset$  for  $e \neq e'$ . Otherwise:  $V^- x = x$

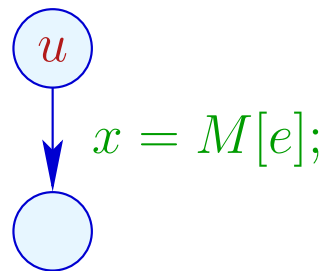
## Transformation CE:



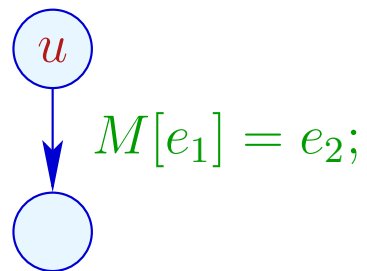
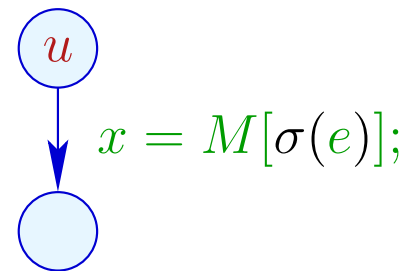
... analogously for edges with  $\text{Neg}(e)$



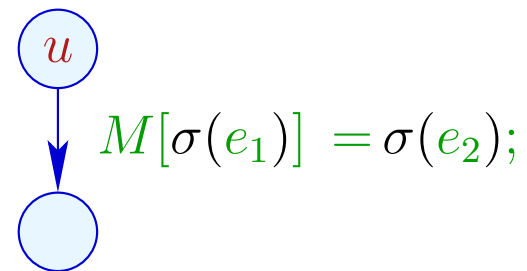
## Transformation CE (cont.):



$$\sigma = \mathcal{V}[u]^-$$



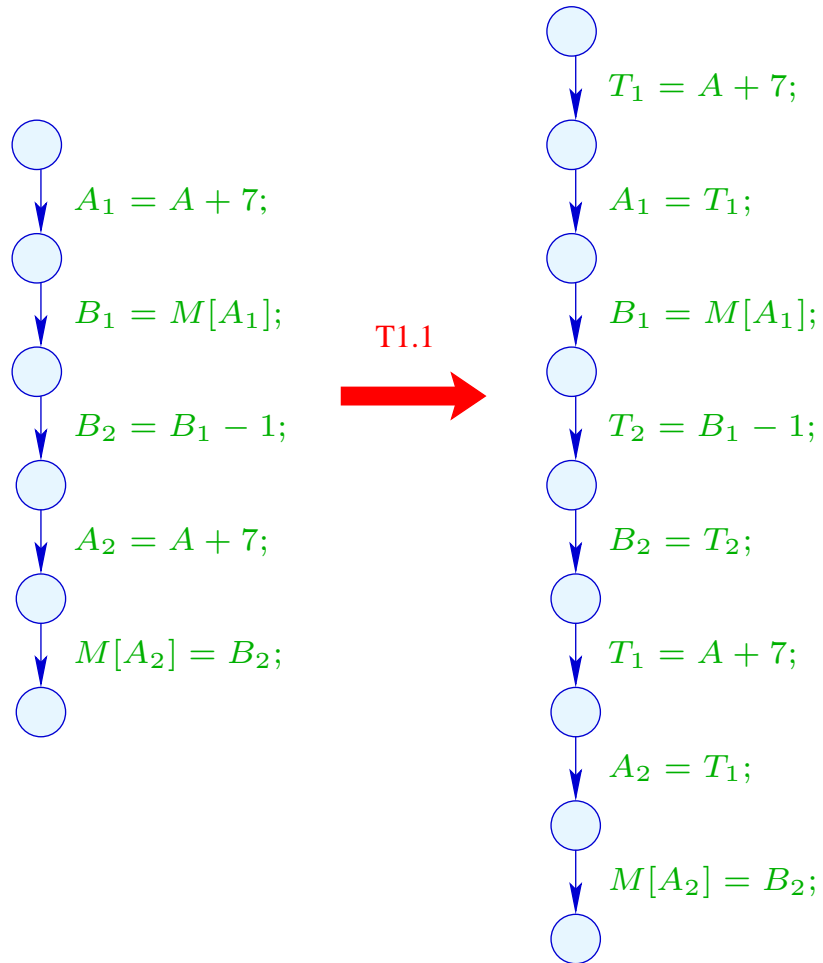
$$\sigma = \mathcal{V}[u]^-$$



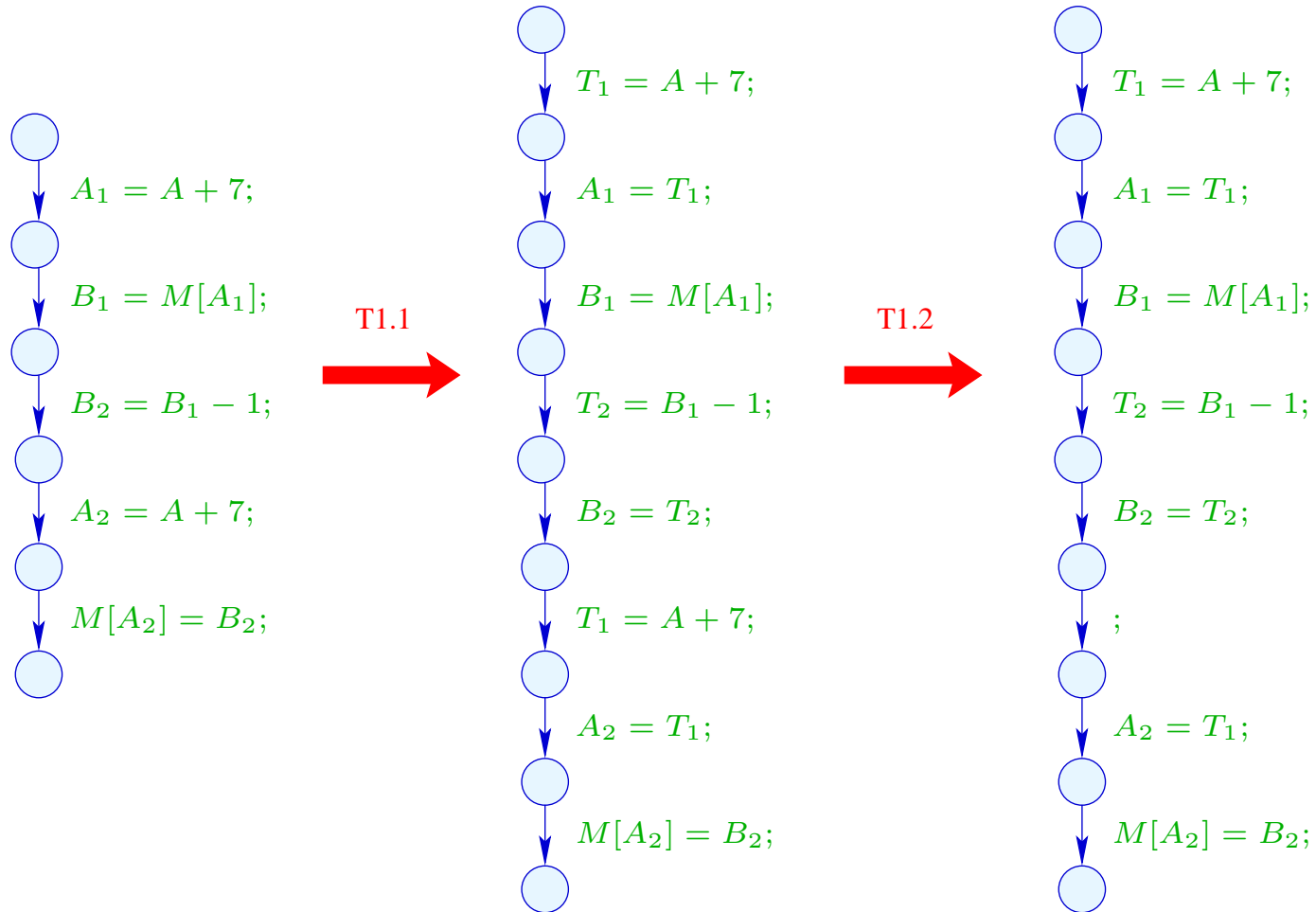
## Procedure as a whole:

- (1) Availability of expressions: T1
  - + removes arithmetic operations
  - inserts superfluous moves
  
- (2) Values of variables: T3
  - + creates dead variables
  
- (3) (true) liveness of variables: T2
  - + removes assignments to dead variables

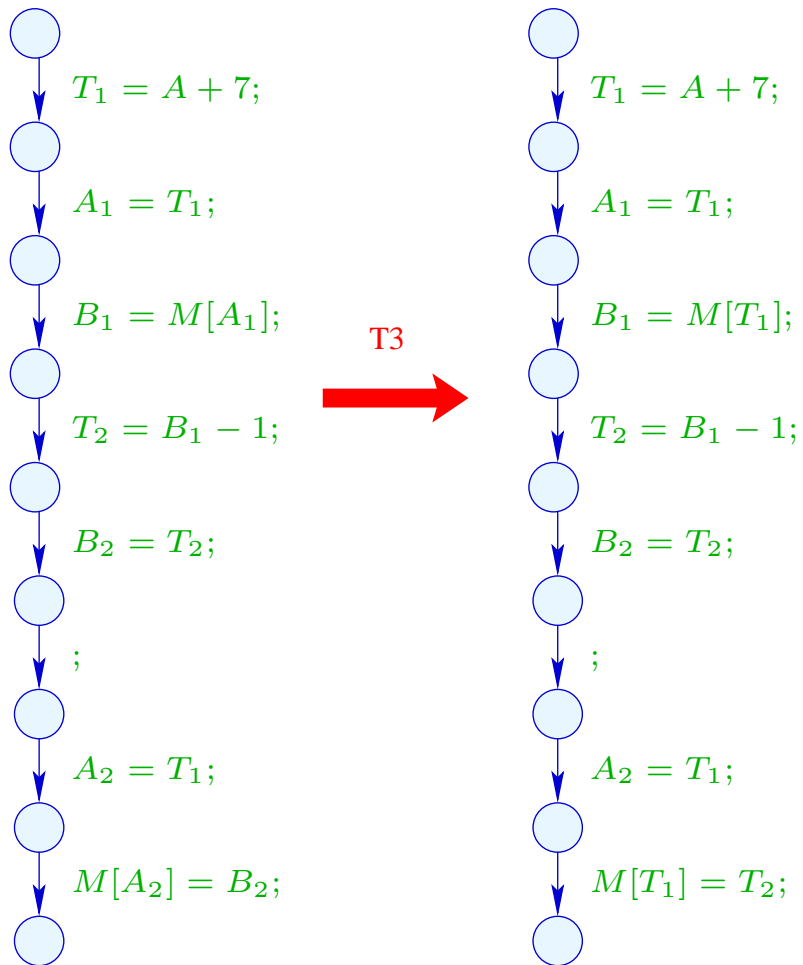
Example: `a[7]--;`



Example: `a[7]--;`



Example (cont.):  $a[7]--i$



Example (cont.):  $a[7]--i$

