

## 1.2 Beseitigung überflüssiger Zuweisungen

Beispiel:

1 :  $x = y + 2;$

2 :  $y = 5;$

3 :  $x = y + 3;$

Der Wert von  $x$  an den Programmpunkten 1, 2 wird überschrieben, bevor er benutzt werden kann.

Die Variable  $x$  nennen wir deshalb an diesen Programmpunkten **tot** :-)

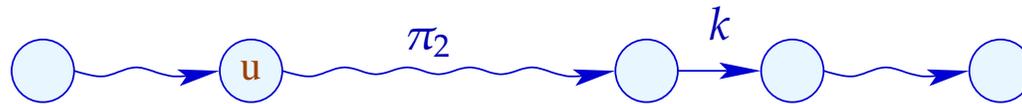
## Beachte:

- Zuweisungen an tote Variablen können wir uns schenken ;-)
- Solche Ineffizienzen können u.a. durch andere Transformationen hervorgerufen werden.

## Formale Definition:

Die Variable  $x$  heißt **lebendig** an  $u$  entlang des Pfads  $\pi$ , falls sich  $\pi$  zerlegen lässt in  $\pi = \pi_1 \pi_2 k \pi_3$  so dass gilt:

- $\pi_1$  erreicht  $u$ ;
- $k$  ist eine **Benutzung** von  $x$ ;
- $\pi_2$  enthält keine **Überschreibung** von  $x$ .

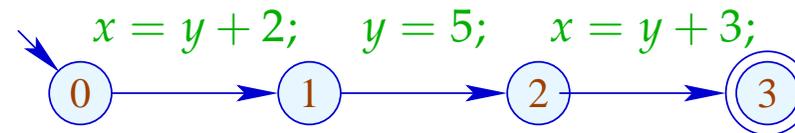


Die Menge der an einer Kante  $k = (\_, lab, \_)$  benutzten bzw. überschriebenen Variablen ist dabei gegeben durch:

<i>lab</i>	benutzt	überschrieben
<i>;</i>	$\emptyset$	$\emptyset$
<i>Pos (e)</i>	<i>Vars (e)</i>	$\emptyset$
<i>Neg (e)</i>	<i>Vars (e)</i>	$\emptyset$
<i>R = e;</i>	<i>Vars (e)</i>	$\{R\}$
<i>R<sub>1</sub> = M[R<sub>2</sub>];</i>	$\{R_2\}$	$\{R_1\}$
<i>M[R<sub>1</sub>] = R<sub>2</sub>;</i>	$\{R_1, R_2\}$	$\emptyset$

Eine Variable  $x$ , die nicht lebendig an  $u$  entlang  $\pi$  ist, heißt **tot** an  $u$  entlang  $\pi$ .

Beispiel:



Wir bemerken:

	lebendig	tot
0	{ $y$ }	{ $x$ }
1	$\emptyset$	{ $x, y$ }
2	{ $y$ }	{ $x$ }
3	$\emptyset$	{ $x, y$ }

Die Variable  $x$  ist lebendig an  $u$  falls  $x$  lebendig ist an  $u$  entlang irgend eines Pfads. Andernfalls ist  $x$  tot an  $u$ .

Die Variable  $x$  ist lebendig an  $u$  falls  $x$  lebendig ist an  $u$  entlang irgend eines Pfads. Andernfalls ist  $x$  tot an  $u$ .

Frage:

Wie berechnet man für jedes  $u$  die Menge der dort lebendigen/toten Variablen ???

Die Variable  $x$  ist **lebendig** an  $u$  falls  $x$  lebendig ist an  $u$  entlang **irgend eines** Pfads. Andernfalls ist  $x$  **tot** an  $u$ .

**Frage:**

Wie berechnet man für jedes  $u$  die Menge der dort lebendigen/toten Variablen ???

**Idee:**

Definiere für jede Kante  $k = (u, \_, v)$  eine Funktion  $[[k]]^\#$ , die die Menge der an  $v$  lebendigen Variablen in die Menge der an  $u$  lebendigen Variablen transformiert ...

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

Für  $k = (\_, lab, \_)$  definieren wir  $\llbracket k \rrbracket^\# = \llbracket lab \rrbracket^\#$  durch:

$$\begin{aligned}\llbracket ; \rrbracket^\# L &= L \\ \llbracket \mathbf{Pos}(e) \rrbracket^\# L &= \llbracket \mathbf{Neg}(e) \rrbracket^\# L = L \cup Vars(e) \\ \llbracket x = e; \rrbracket^\# L &= (L \setminus \{x\}) \cup Vars(e) \\ \llbracket R_1 = M[R_2]; \rrbracket^\# L &= (L \setminus \{R_1\}) \cup \{R_2\} \\ \llbracket M[R_1] = R_2; \rrbracket^\# L &= L \cup \{R_1, R_2\}\end{aligned}$$

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

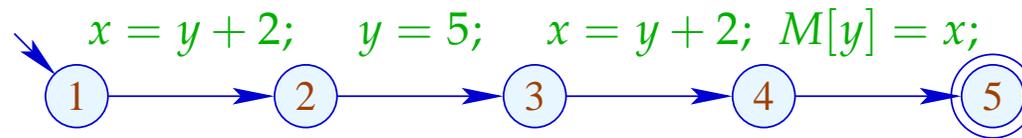
Für  $k = (\_, lab, \_)$  definieren wir  $\llbracket k \rrbracket^\# = \llbracket lab \rrbracket^\#$  durch:

$$\begin{aligned}\llbracket ; \rrbracket^\# L &= L \\ \llbracket \mathbf{Pos}(e) \rrbracket^\# L &= \llbracket \mathbf{Neg}(e) \rrbracket^\# L = L \cup Vars(e) \\ \llbracket x = e; \rrbracket^\# L &= (L \setminus \{x\}) \cup Vars(e) \\ \llbracket R_1 = M[R_2]; \rrbracket^\# L &= (L \setminus \{R_1\}) \cup \{R_2\} \\ \llbracket M[R_1] = R_2; \rrbracket^\# L &= L \cup \{R_1, R_2\}\end{aligned}$$

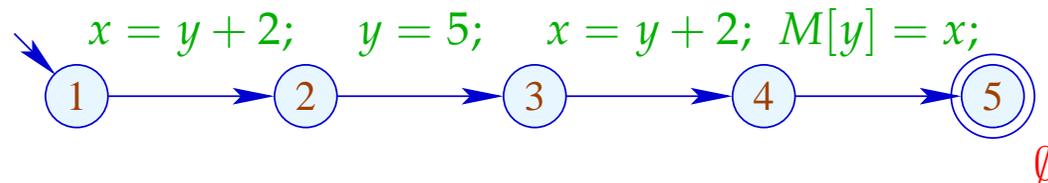
$\llbracket k \rrbracket^\#$  können wir wieder zu Effekten  $\llbracket \pi \rrbracket^\#$  ganzer Pfade  $\pi = k_1 \dots k_r$  fortsetzen durch:

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

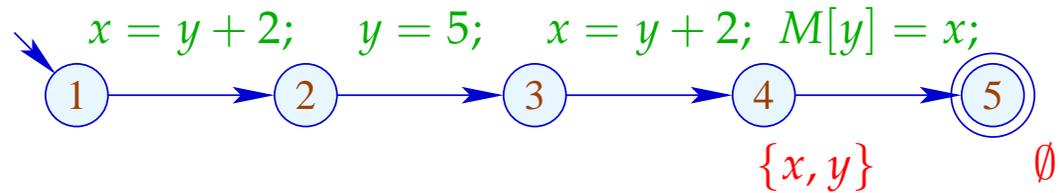
Wir vergewissern uns, dass diese Definitionen vernünftig sind :-)



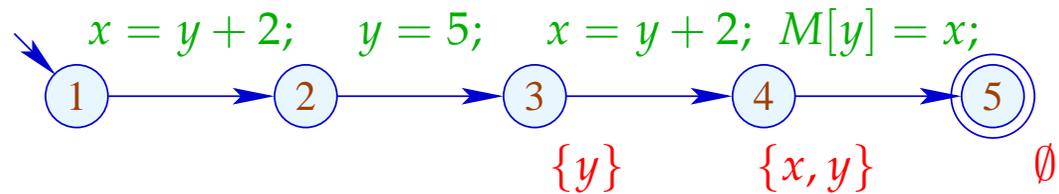
Wir vergewissern uns, dass diese Definitionen vernünftig sind :-)



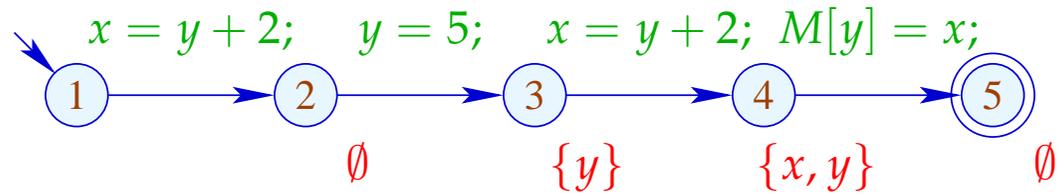
Wir vergewissern uns, dass diese Definitionen vernünftig sind :-)



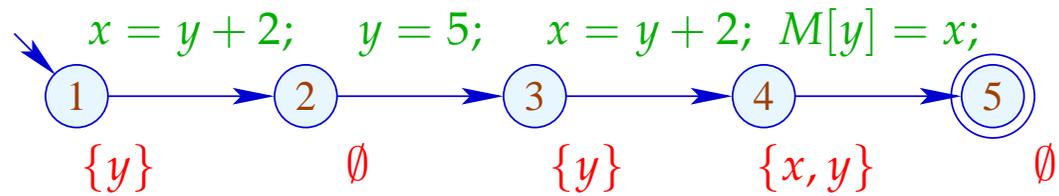
Wir vergewissern uns, dass diese Definitionen vernünftig sind :-)



Wir vergewissern uns, dass diese Definitionen vernünftig sind :-)



Wir vergewissern uns, dass diese Definitionen vernünftig sind :-)



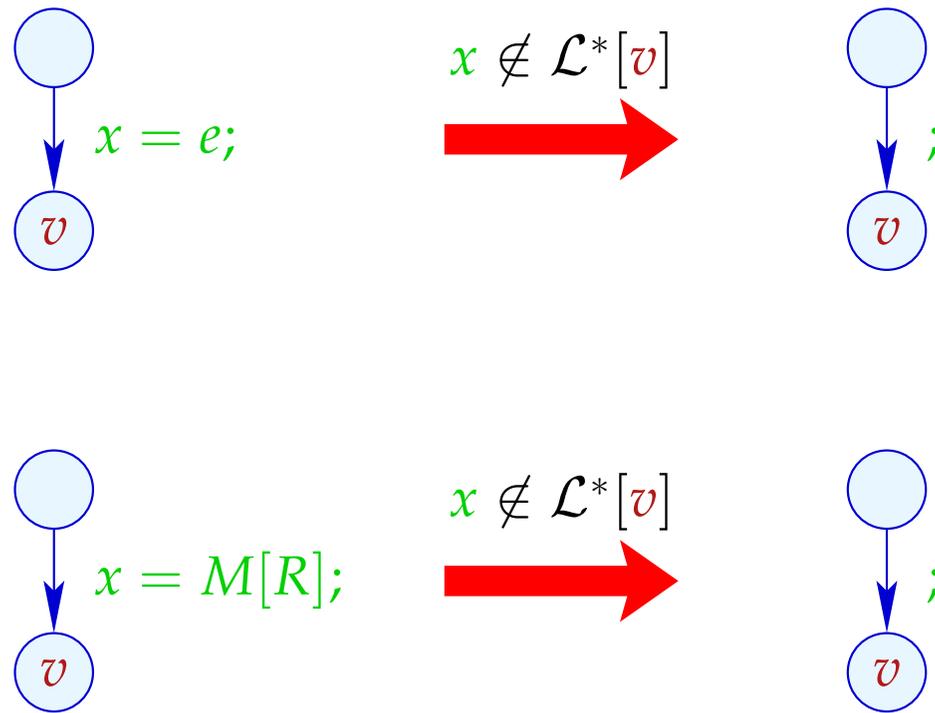
Die Menge der an  $u$  lebendigen Variablen ist dann:

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

... in Worten:

- Die Pfade **starten** in  $u$  :-)
- $x$  ist lebendig, wenn es nur entlang irgend eines Pfads lebendig ist :-)
- $\implies$  Als Halbordnung für  $\mathbb{L}$  benötigen wir  $\sqsubseteq = \subseteq$ .
- Am Programmende ist **keine** Variable mehr lebendig :-)

### Transformation 3:



## Zur Korrektheit zeigt man:

- **Korrektheit der Kanten-Effekte:** Falls  $L$  die Menge der lebendigen Variablen am Ende eines Pfads  $\pi$  sind, dann ist  $\llbracket \pi \rrbracket^\# L$  die Menge der am Anfang lebendigen Variablen :-)
- **Korrektheit der Transformation auf einem Pfad:** Wird auf den Wert einer Variable zugegriffen, ist diese stets lebendig. Der Wert toter Variablen ist darum egal :-)
- **Korrektheit der Transformation:** Bei Ausführung des transformierten Programms haben bei jedem Besuch eines Programmpunkts die lebendigen Variablen den gleichen Wert :-))

## Berechnung der Mengen $\mathcal{L}^*[u]$ :

(1) Aufstellen des Constraint-Systems:

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

(2) Lösen des Constraint-Systems mittels RR-Iteration.

Da  $\mathbb{L}$  endlich ist, terminiert die Iteration :-)

(3) Die kleinste Lösung  $\mathcal{L}$  des Constraint-Systems ist gleich  $\mathcal{L}^*$  da alle  $\llbracket k \rrbracket^\#$  distributiv sind :-))

## Berechnung der Mengen $\mathcal{L}^*[u]$ :

(1) Aufstellen des Constraint-Systems:

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

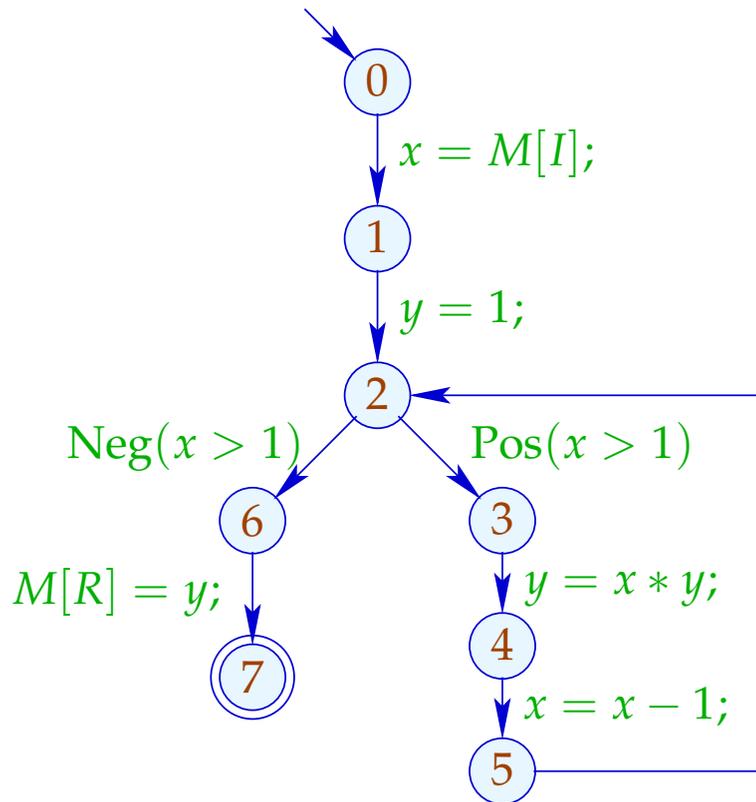
(2) Lösen des Constraint-Systems mittels RR-Iteration.

Da  $\mathbb{L}$  endlich ist, terminiert die Iteration :-)

(3) Die kleinste Lösung  $\mathcal{L}$  des Constraint-Systems ist gleich  $\mathcal{L}^*$  da alle  $\llbracket k \rrbracket^\#$  distributiv sind :-))

**Achtung:** Die Information wird rückwärts propagiert !!!

## Beispiel:



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

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

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

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

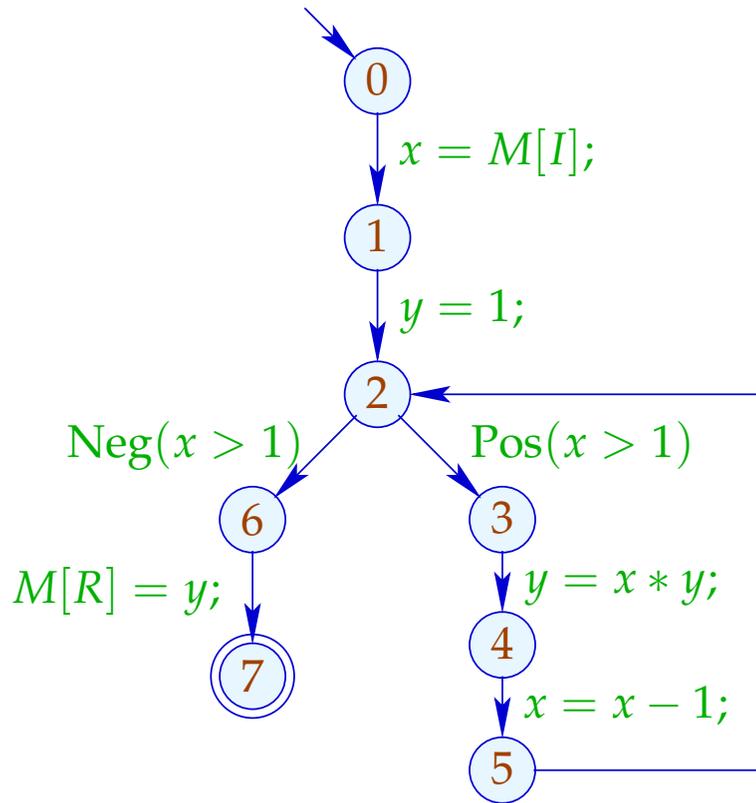
$$\mathcal{L}[4] \supseteq (\mathcal{L}[5] \setminus \{x\}) \cup \{x\}$$

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

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

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

# Beispiel:

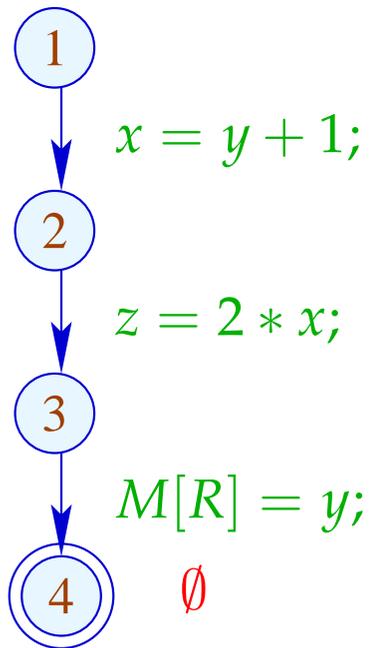


	1	2
7	$\emptyset$	
6	$\{y, R\}$	
2	$\{x, y, R\}$	dito
5	$\{x, y, R\}$	
4	$\{x, y, R\}$	
3	$\{x, y, R\}$	
1	$\{x, R\}$	
0	$\{I, R\}$	

Bei keiner Zuweisung ist die linke Variable **tot** :-)

## Achtung:

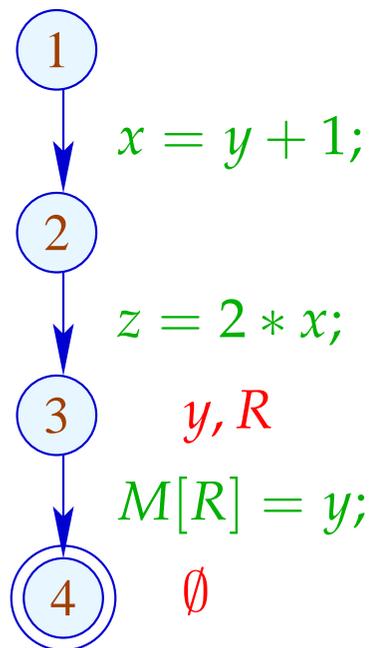
Beseitigung von Zuweisungen an tote Variablen kann weitere Variablen töten:



Bei keiner Zuweisung ist die linke Variable **tot** :-)

## Achtung:

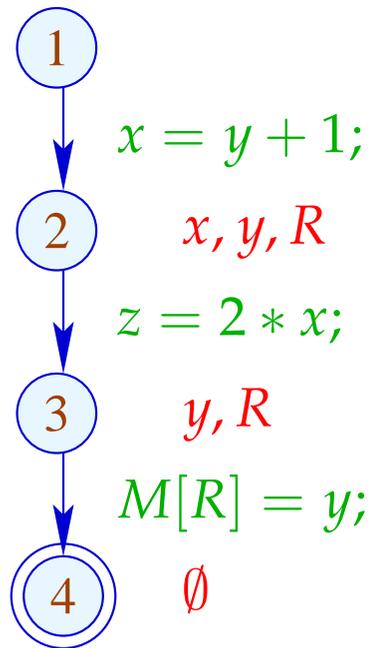
Beseitigung von Zuweisungen an tote Variablen kann weitere Variablen töten:



Bei keiner Zuweisung ist die linke Variable **tot** :-)

## Achtung:

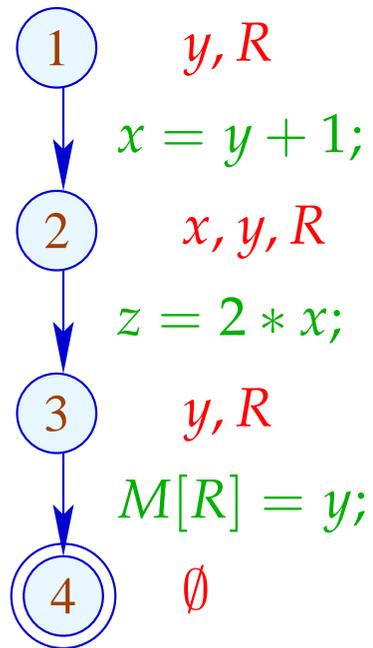
Beseitigung von Zuweisungen an tote Variablen kann weitere Variablen töten:



Bei keiner Zuweisung ist die linke Variable **tot** :-)

## Achtung:

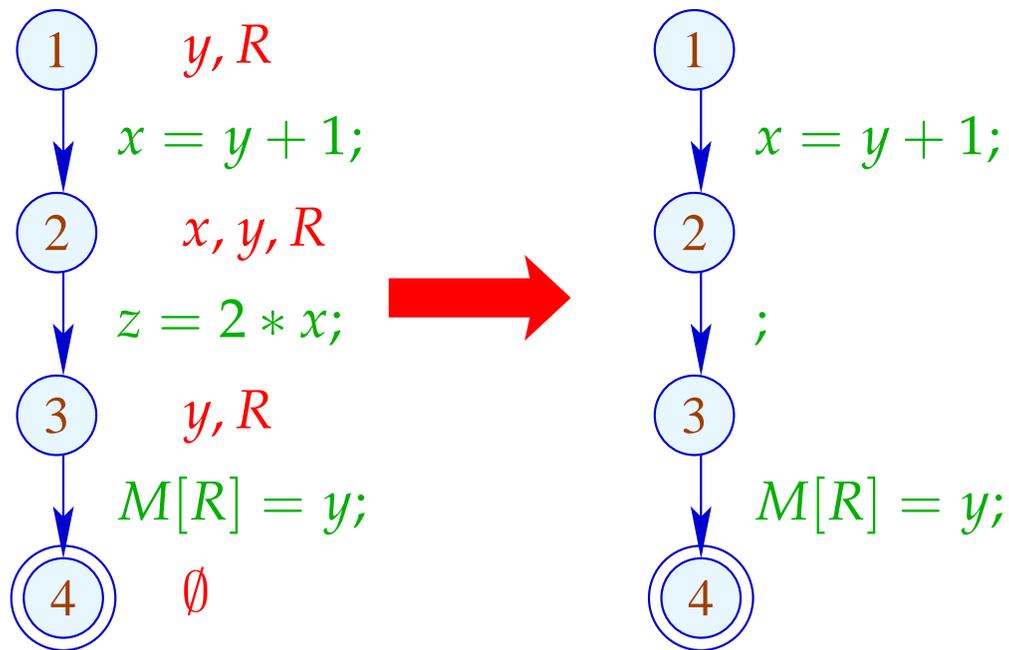
Beseitigung von Zuweisungen an tote Variablen kann weitere Variablen töten:



Bei keiner Zuweisung ist die linke Variable **tot** :-)

## Achtung:

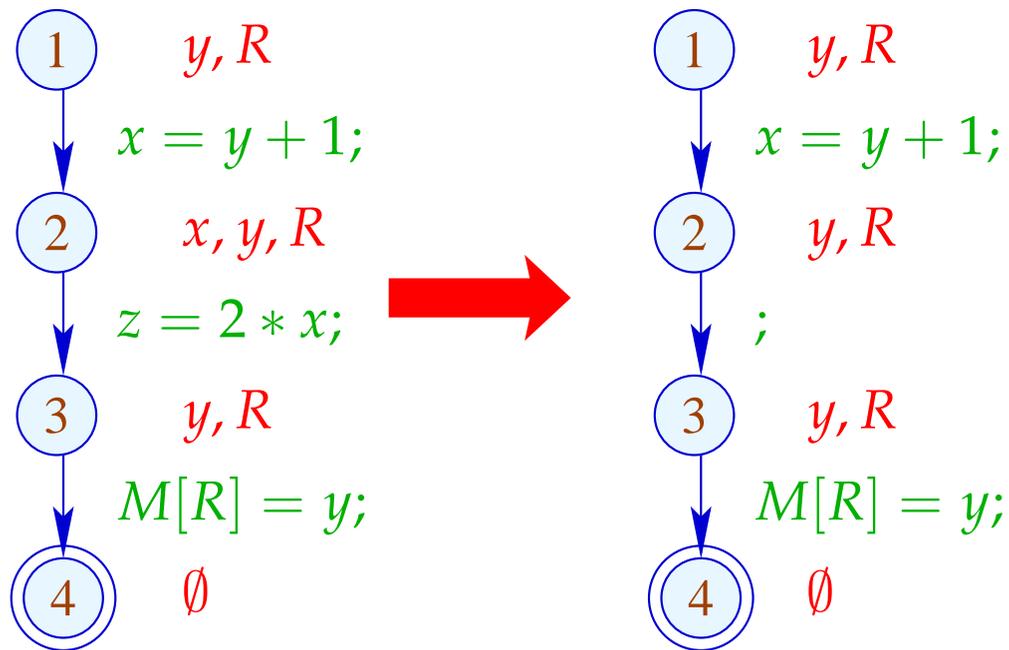
Beseitigung von Zuweisungen an tote Variablen kann weitere Variablen töten:



Bei keiner Zuweisung ist die linke Variable **tot** :-)

## Achtung:

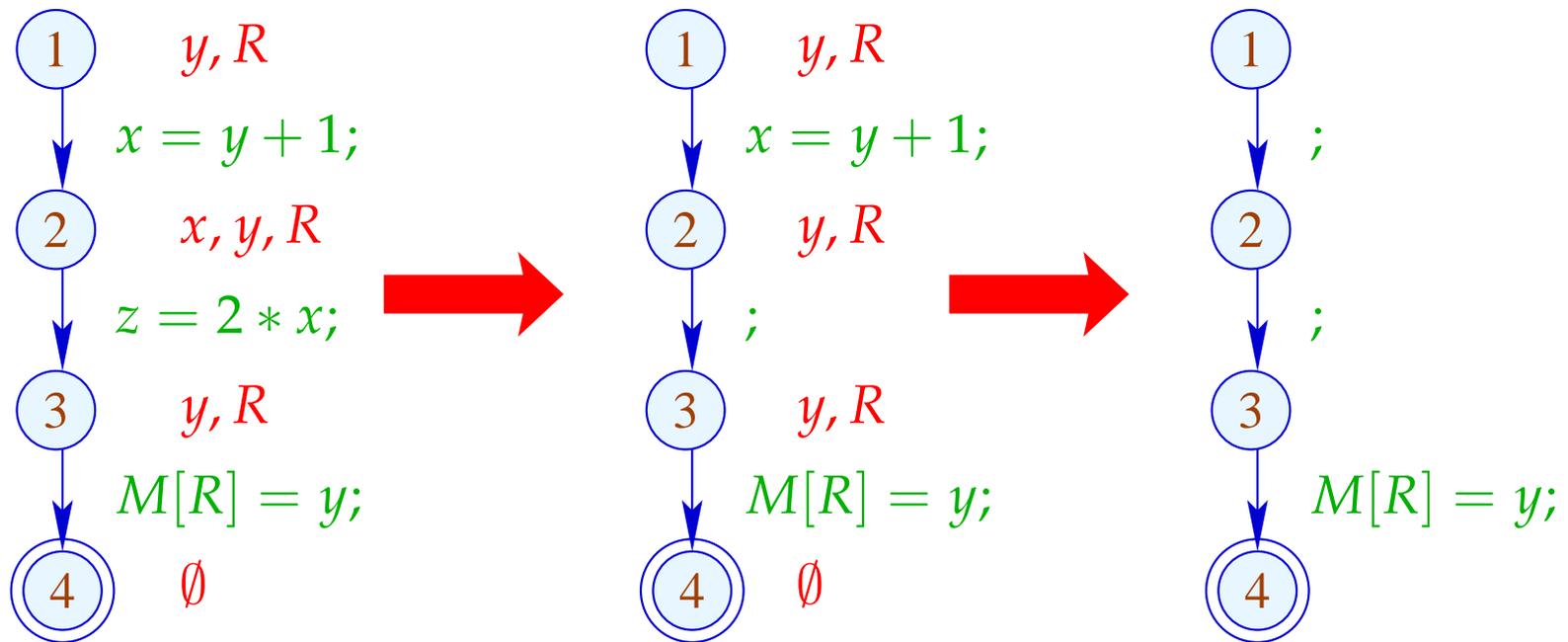
Beseitigung von Zuweisungen an tote Variablen kann weitere Variablen töten:



Bei keiner Zuweisung ist die linke Variable **tot** :-)

## Achtung:

Beseitigung von Zuweisungen an tote Variablen kann weitere Variablen töten:



Das Programm mehrmals zu analysieren, ist hässlich :-)

Idee: Analysiere **echte** Lebendigkeit!

$x$  heißt **echt lebendig** an  $u$  entlang eines Pfads  $\pi$ , falls sich  $\pi$  zerlegen lässt in  $\pi = \pi_1 \pi_2 k \pi_3$  so dass gilt:

- $\pi_1$  erreicht  $u$ ;
- $k$  ist eine **echte** Benutzung von  $x$ ;
- $\pi_2$  enthält keine **Überschreibung** von  $x$ .

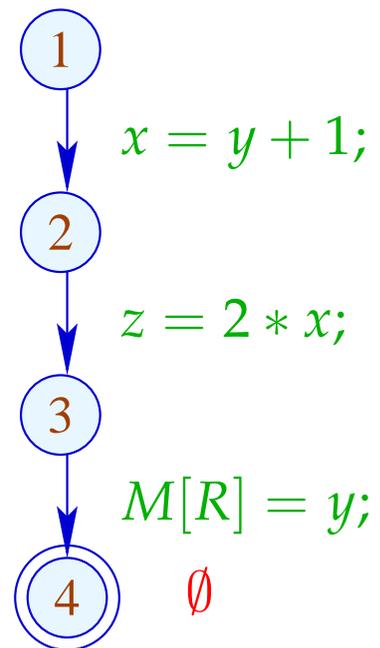


Die Menge der an einer Kante  $k = (\_, lab, v)$  echt benutzten Variablen ist gegeben durch:

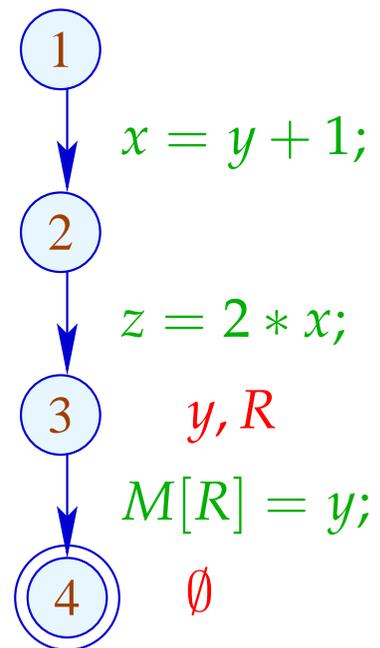
<i>lab</i>	echt benutzt
<i>;</i>	$\emptyset$
<i>Pos (e)</i>	<i>Vars (e)</i>
<i>Neg (e)</i>	<i>Vars (e)</i>
<i>x = e;</i>	<i>Vars (e)</i> (*)
<i>x = M[R];</i>	$\{R\}$ (*)
<i>M[R<sub>1</sub>] = R<sub>2</sub>;</i>	$\{R_1, R_2\}$

(\*) – sofern  $x$  an  $v$  echt lebendig ist :-)

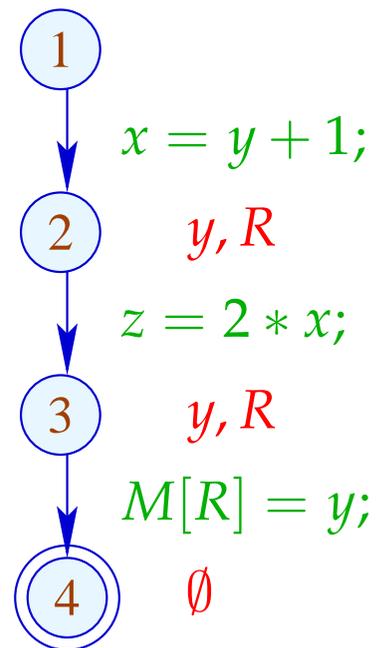
Beispiel:



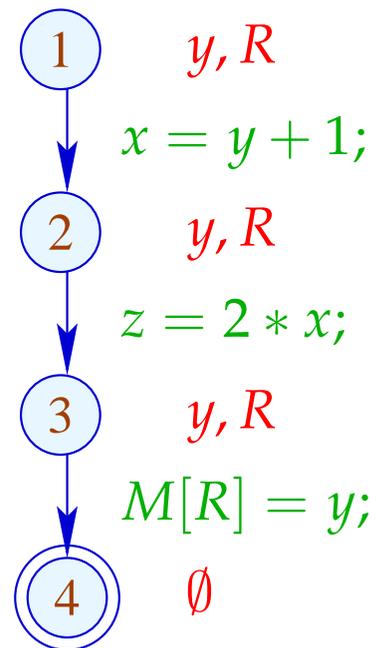
Beispiel:



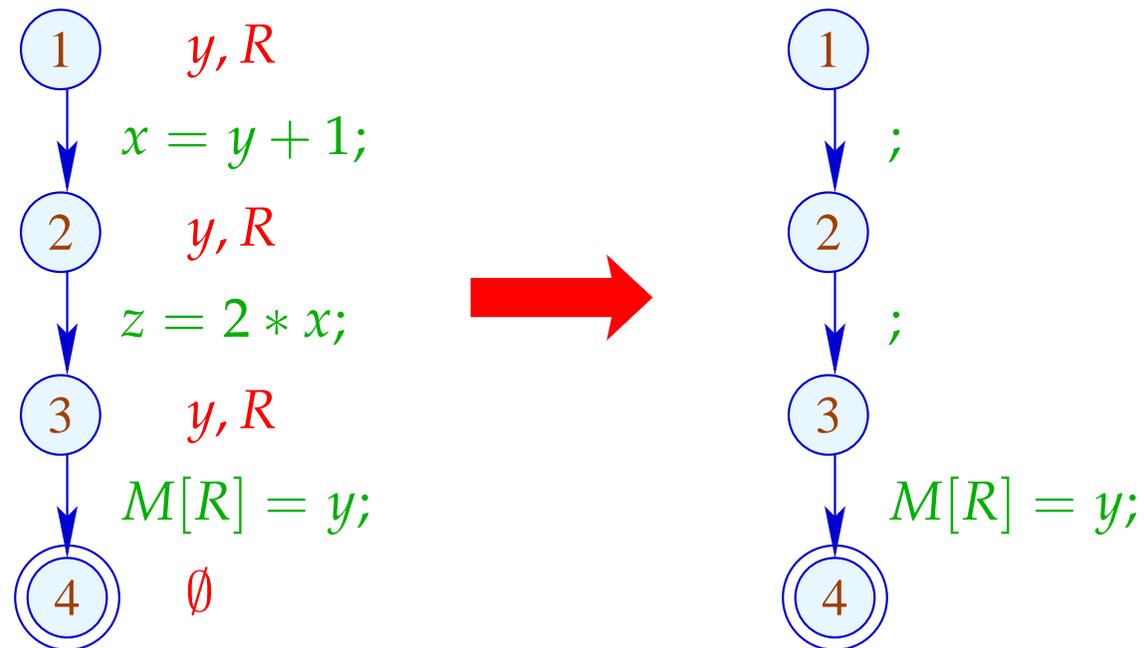
Beispiel:



Beispiel:



## Beispiel:



## Die Kanten-Effekte:

$$\begin{aligned} \llbracket ; \rrbracket^\# L &= L \\ \llbracket \mathbf{Pos}(e) \rrbracket^\# L &= \llbracket \mathbf{Neg}(e) \rrbracket^\# L = L \cup \mathit{Vars}(e) \\ \llbracket x = e; \rrbracket^\# L &= (L \setminus \{x\}) \cup \mathit{Vars}(e) \\ \llbracket R_1 = M[R_2]; \rrbracket^\# L &= (L \setminus \{R_1\}) \cup \{R_2\} \\ \llbracket M[R_1] = R_2; \rrbracket^\# L &= L \cup \{R_1, R_2\} \end{aligned}$$

## Die Kanten-Effekte:

$$\begin{aligned} \llbracket ; \rrbracket^\# L &= L \\ \llbracket \text{Pos}(e) \rrbracket^\# L &= \llbracket \text{Neg}(e) \rrbracket^\# L = L \cup \text{Vars}(e) \\ \llbracket x = e; \rrbracket^\# L &= (L \setminus \{x\}) \cup (x \in L) ? \text{Vars}(e) : \emptyset \\ \llbracket R_1 = M[R_2]; \rrbracket^\# L &= (L \setminus \{R_1\}) \cup (R_1 \in L) ? \{R_2\} : \emptyset \\ \llbracket M[R_1] = R_2; \rrbracket^\# L &= L \cup \{R_1, R_2\} \end{aligned}$$

## Beachte:

- Die Kanten-Effekte für echt lebendige Variablen sind **komplizierter** als für lebendige Variablen :-)
- Sie sind aber immer noch **distributiv !!**

## Beachte:

- Die Kanten-Effekte für echt lebendige Variablen sind **komplizierter** als für lebendige Variablen :-)
- Sie sind aber immer noch **distributiv !!**

Dazu betrachten wir für  $\mathbb{D} = 2^U$ ,  $f y = (u \in y) ? b : \emptyset$

Wir überprüfen:

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

## Beachte:

- Die Kanten-Effekte für echt lebendige Variablen sind **komplizierter** als für lebendige Variablen :-)
- Sie sind aber immer noch **distributiv !!**

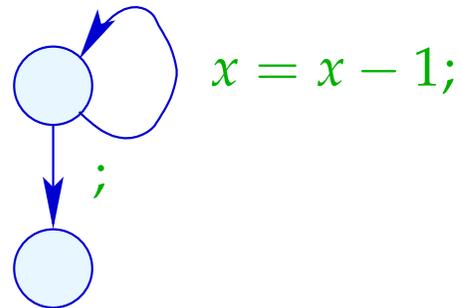
Dazu betrachten wir für  $\mathbb{D} = 2^U$ ,  $f y = (u \in y) ? b : \emptyset$

Wir überprüfen:

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

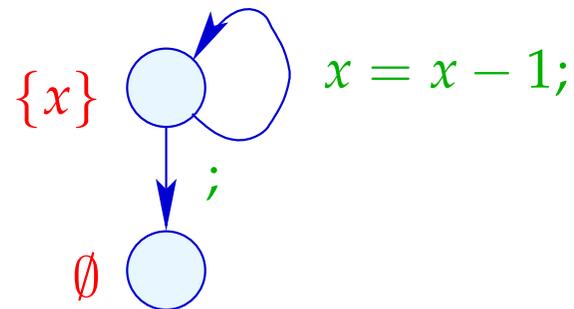
$\implies$  Constraint-System liefert **MOP** :-))

- Echte Lebendigkeit findet **mehr** überflüssige Zuweisungen als wiederholte Lebendigkeit !!!



- Echte Lebendigkeit findet **mehr** überflüssige Zuweisungen als wiederholte Lebendigkeit !!!

Lebendigkeit:



- Echte Lebendigkeit findet **mehr** überflüssige Zuweisungen als wiederholte Lebendigkeit !!!

Echte Lebendigkeit:

