The Effects of Edges:

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

The Effects of Edges:

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

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:

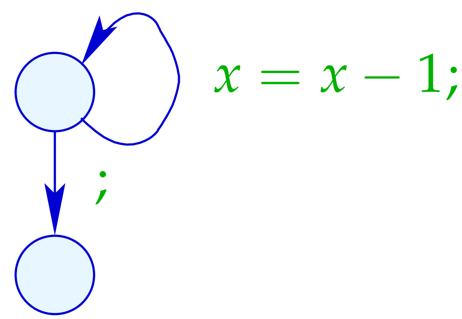
- The effects of edges for truely live variables are more complicated than for live variables :-)
- Nonetheless, they are distributive !!

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

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

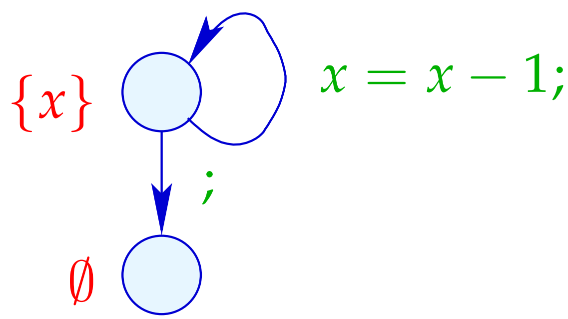
$\Longrightarrow$ the constraint system yields the MOP :-))

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



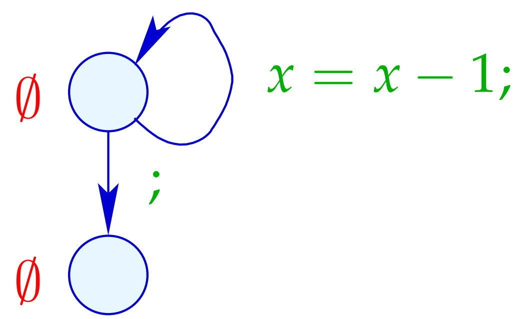
- True liveness detects more superfluous assignments than repeated liveness !!!

Liveness:



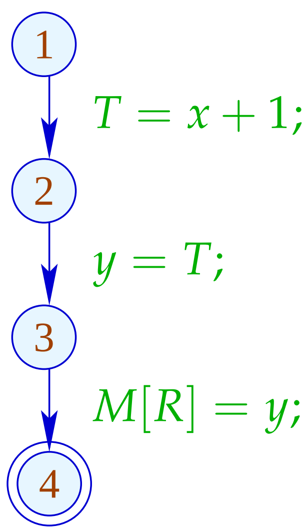
- True liveness detects more superfluous assignments than repeated liveness !!!

True Liveness:



1.3 Removing Superfluous Moves

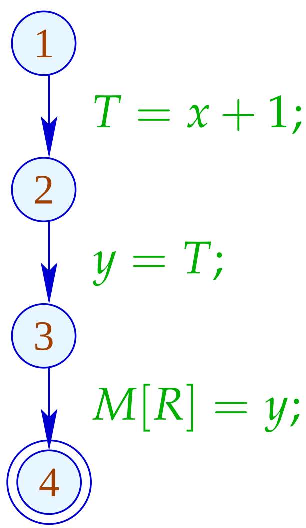
Example:



This variable-variable assignment is obviously useless :-(

1.3 Removing Superfluous Moves

Example:

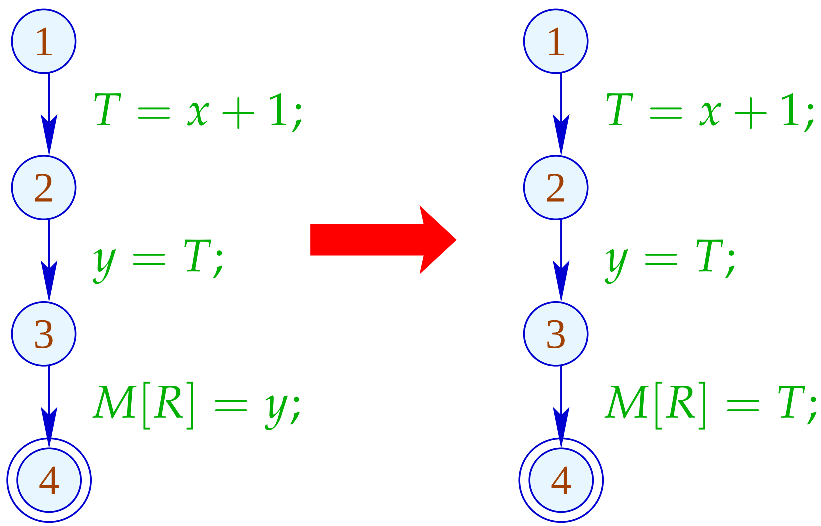


This variable-variable assignment is obviously useless :-(

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

1.3 Removing Superfluous Moves

Example:

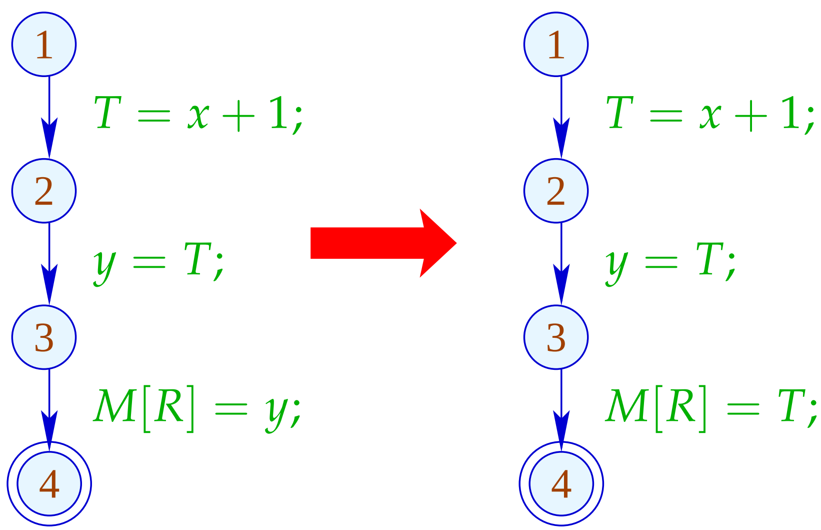


This variable-variable assignment is obviously useless :-(

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

1.3 Removing Superfluous Moves

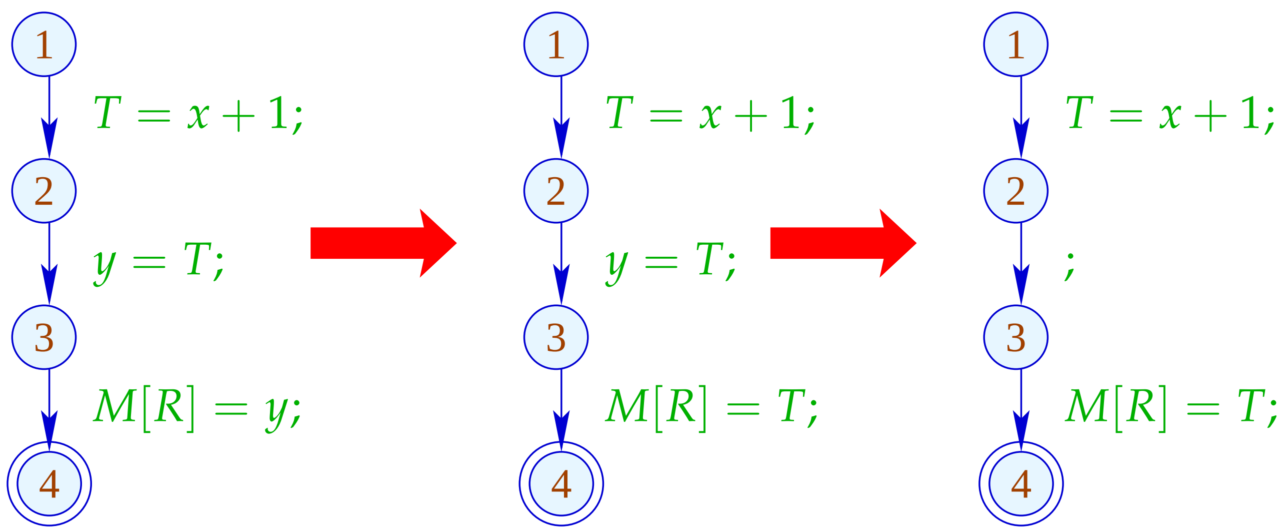
Example:



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 :-)

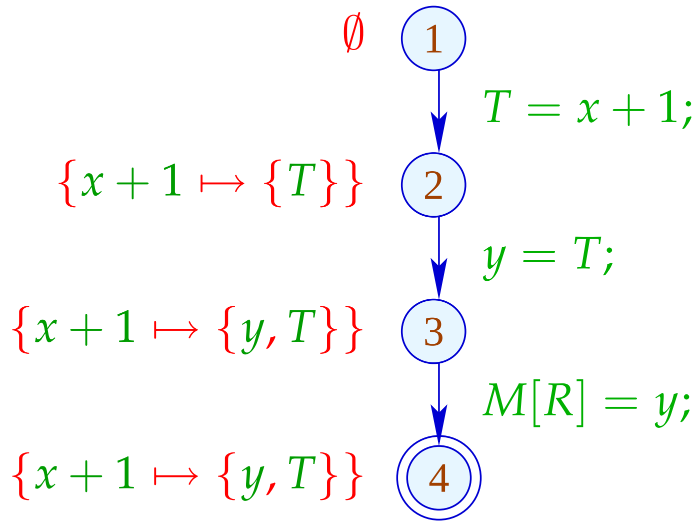
We use: $\mathbb{V} = \mathit{Expr} \to 2^{\mathit{Vars}}$ and define:

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

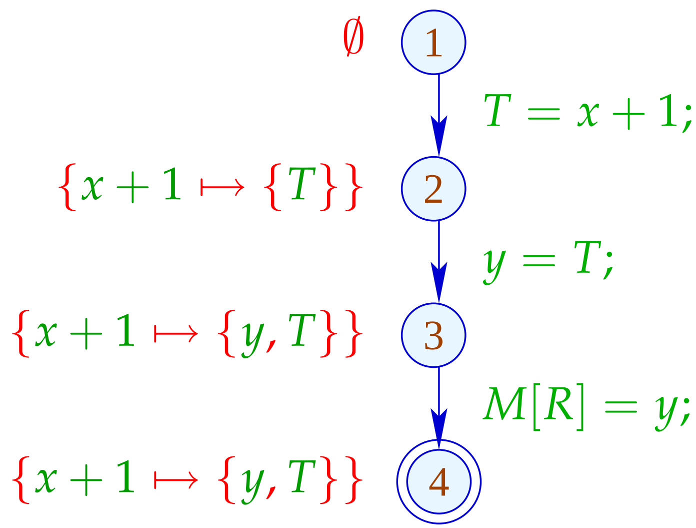
$$
\begin{array}{lcl}
[\![x = c;]\!]^\sharp\, V\, e' & = & \begin{cases} (V\, c) \cup \{x\} & \text{if} \quad e' = c \\ (V\, e') \backslash \{x\} & \text{otherwise} \end{cases} \\
[\![x = y;]\!]^\sharp\, V\, e & = & \begin{cases} (V\, e) \cup \{x\} & \text{if} \quad y \in V\, e \\ (V\, e) \backslash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

// analogously for the diverse stores

In the Example:



In the Example:



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

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

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

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

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:

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

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

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

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

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

$\Longrightarrow$

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: $V^- \, x = x$:-)

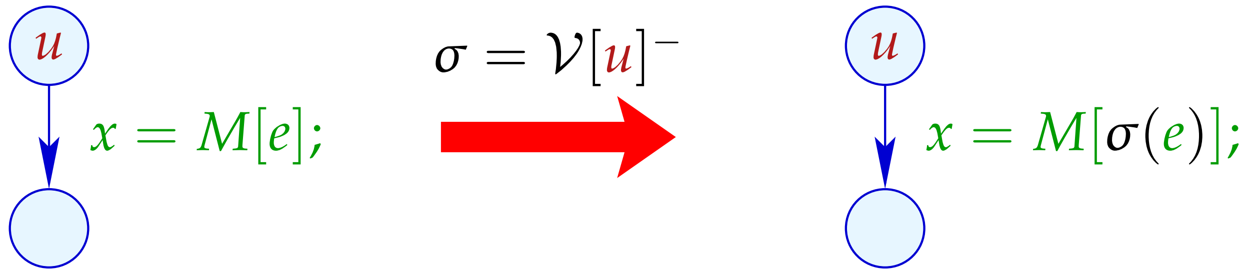
Transformation 3:

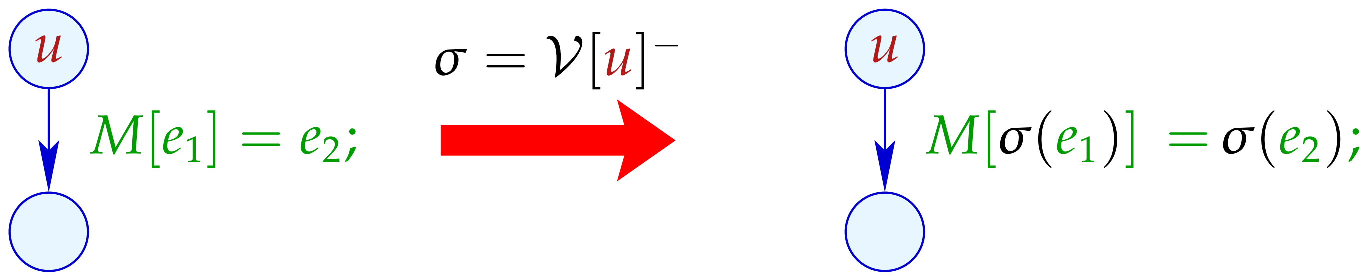
$$\begin{tikzcd} u \arrow[d, "\text{Pos}\,(e)"] & & u \arrow[d, "\text{Pos}\,(\sigma(e))"] \\ {} \arrow[r, phantom, ""] & {} \arrow[r, Rightarrow, "\sigma = \mathcal{V}[u]^{-}"] & {} \end{tikzcd}$$

... analogously for edges with Neg (e)

$$\begin{tikzcd} u \arrow[d, "x = e;"] & & u \arrow[d, "x = \sigma(e);"] \\ {} \arrow[r, phantom, ""] & {} \arrow[r, Rightarrow, "\sigma = \mathcal{V}[u]^{-}"] & {} \end{tikzcd}$$

Transformation 3 (cont.):





Procedure as a whole:

(1) Availability of expressions: T1

- \+ removes arithmetic operations
- − inserts superfluous moves

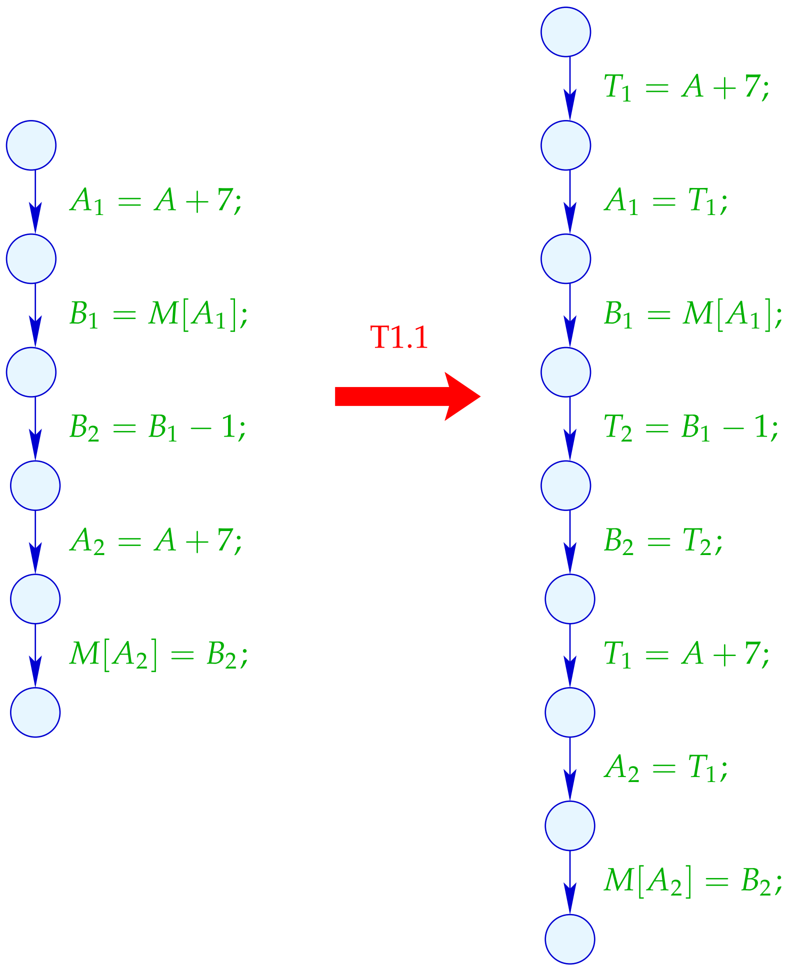
(2) Values of variables: T3

- \+ creates dead variables

(3) (true) liveness of variables: T2

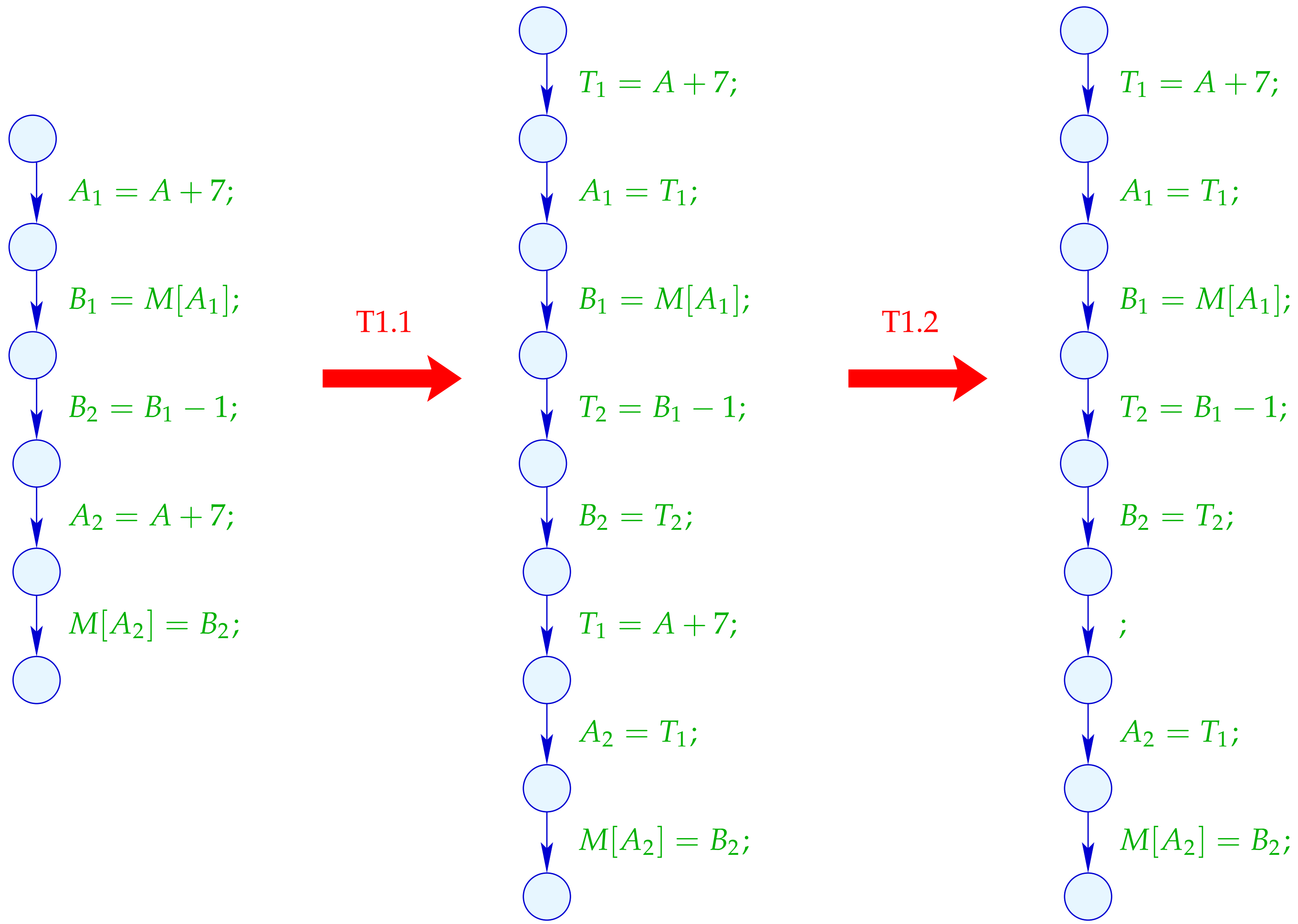
- \+ removes assignments to dead variables

Example: `a[7]--;`



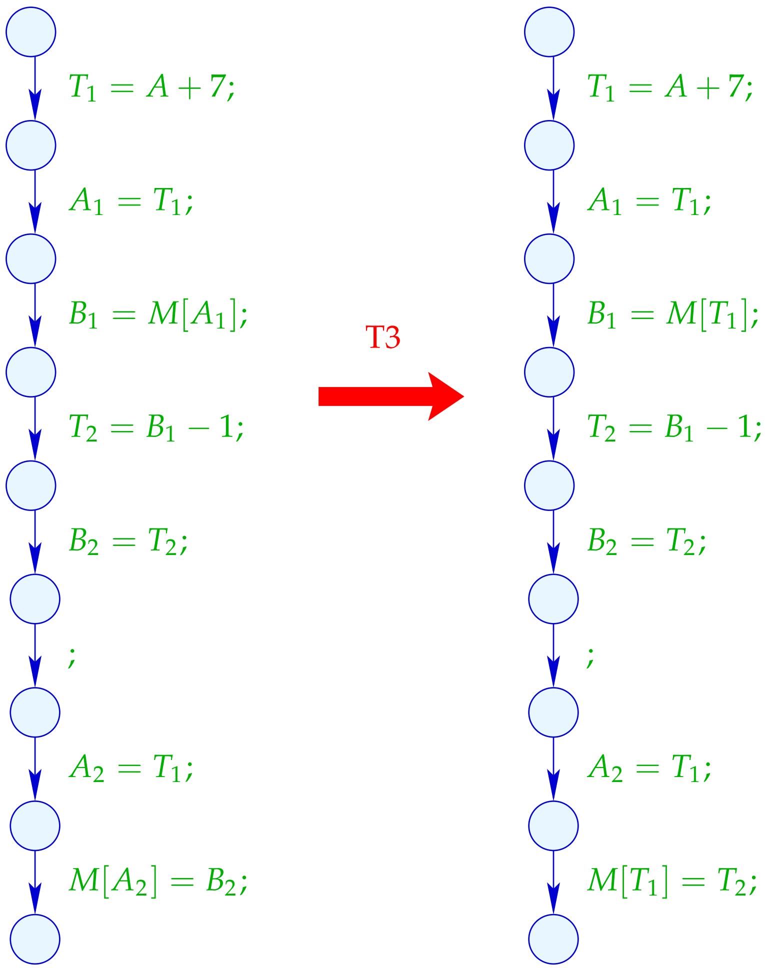
$A_1 = A + 7;$
$B_1 = M[A_1];$
$B_2 = B_1 - 1;$
$A_2 = A + 7;$
$M[A_2] = B_2;$
T1.1
$T_1 = A + 7;$
$A_1 = T_1;$
$B_1 = M[A_1];$
$T_2 = B_1 - 1;$
$B_2 = T_2;$
$T_1 = A + 7;$
$A_2 = T_1;$
$M[A_2] = B_2;$

Example: `a[7]--;`

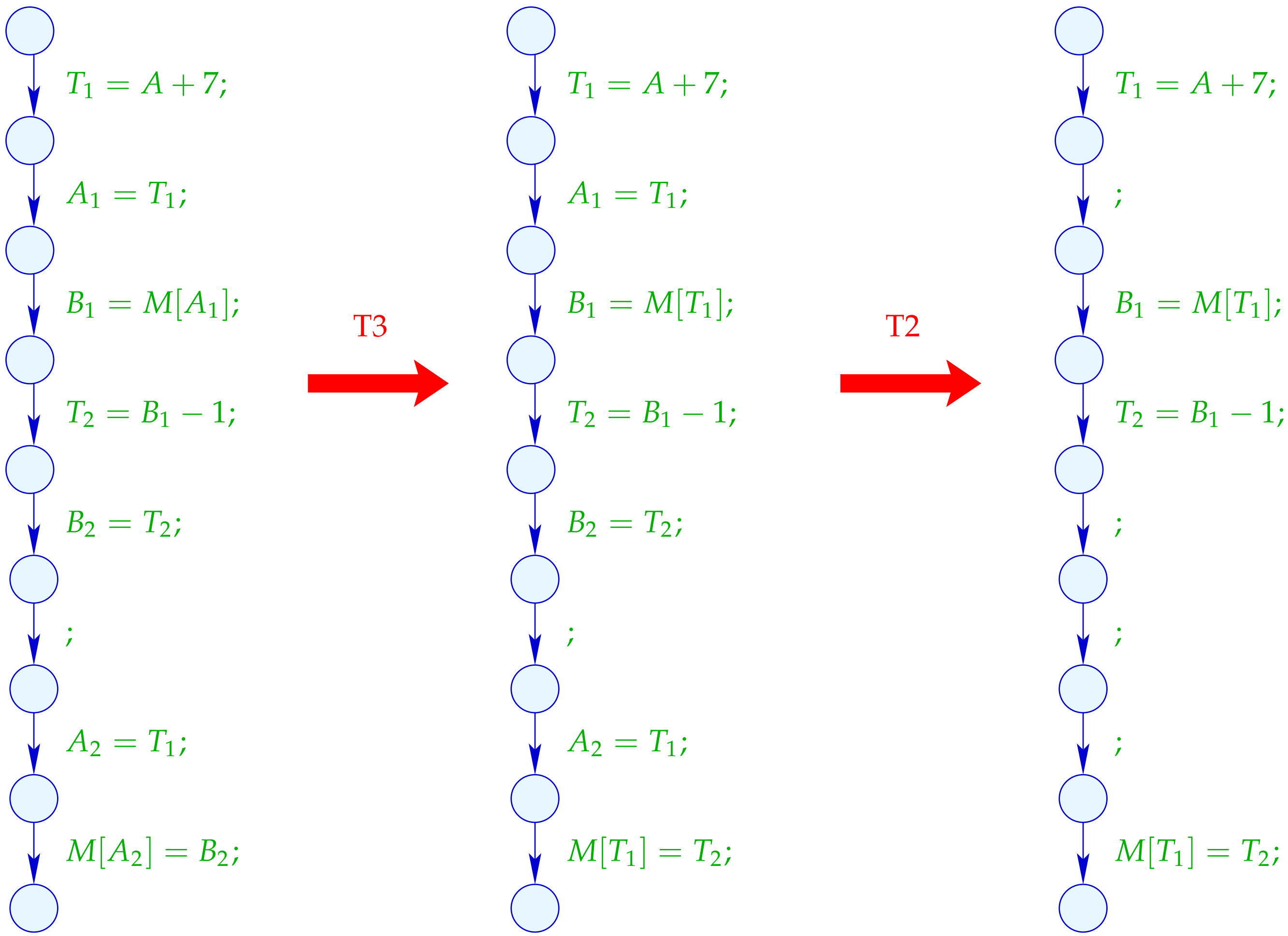


$A_1 = A + 7;$
$B_1 = M[A_1];$
$B_2 = B_1 - 1;$
$A_2 = A + 7;$
$M[A_2] = B_2;$
T1.1
$T_1 = A + 7;$
$A_1 = T_1;$
$B_1 = M[A_1];$
$T_2 = B_1 - 1;$
$B_2 = T_2;$
$T_1 = A + 7;$
$A_2 = T_1;$
$M[A_2] = B_2;$
T1.2
$T_1 = A + 7;$
$A_1 = T_1;$
$B_1 = M[A_1];$
$T_2 = B_1 - 1;$
$B_2 = T_2;$
$;$
$A_2 = T_1;$
$M[A_2] = B_2;$

Example (cont.): `a[7]--;`



Example (cont.): `a[7]--;`



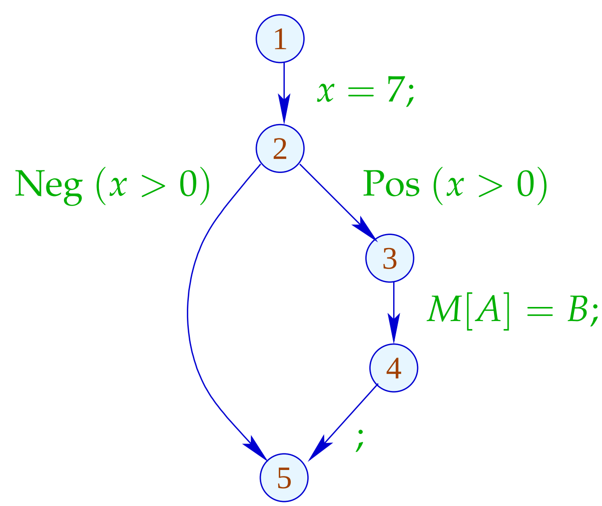
1.4 Constant Propagation

Idea:

Execute as much of the code at compile-time as possible!

Example:

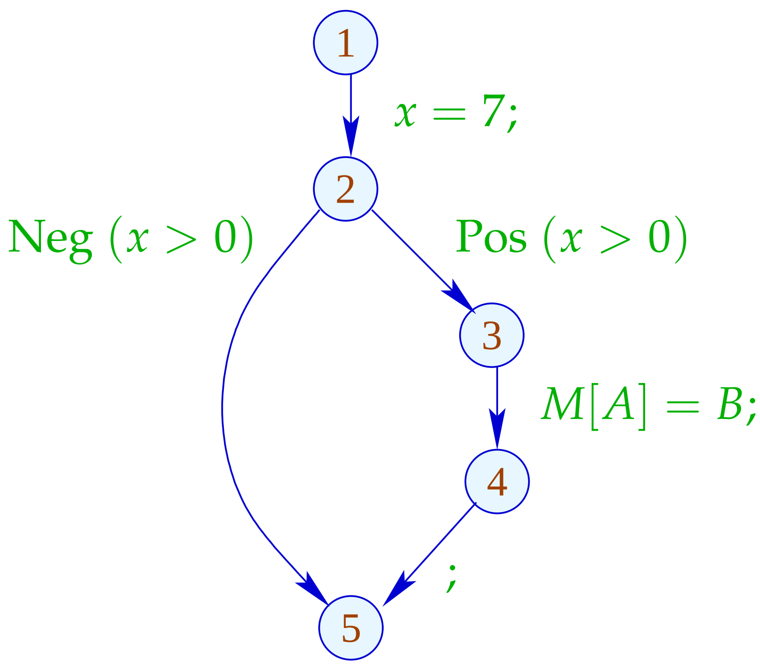
$$x = 7;$$
$$\text{if } (x > 0)$$
$$\qquad M[A] = B;$$



Obviously, x has always the value 7 :-)

Thus, the memory access is always executed :-))

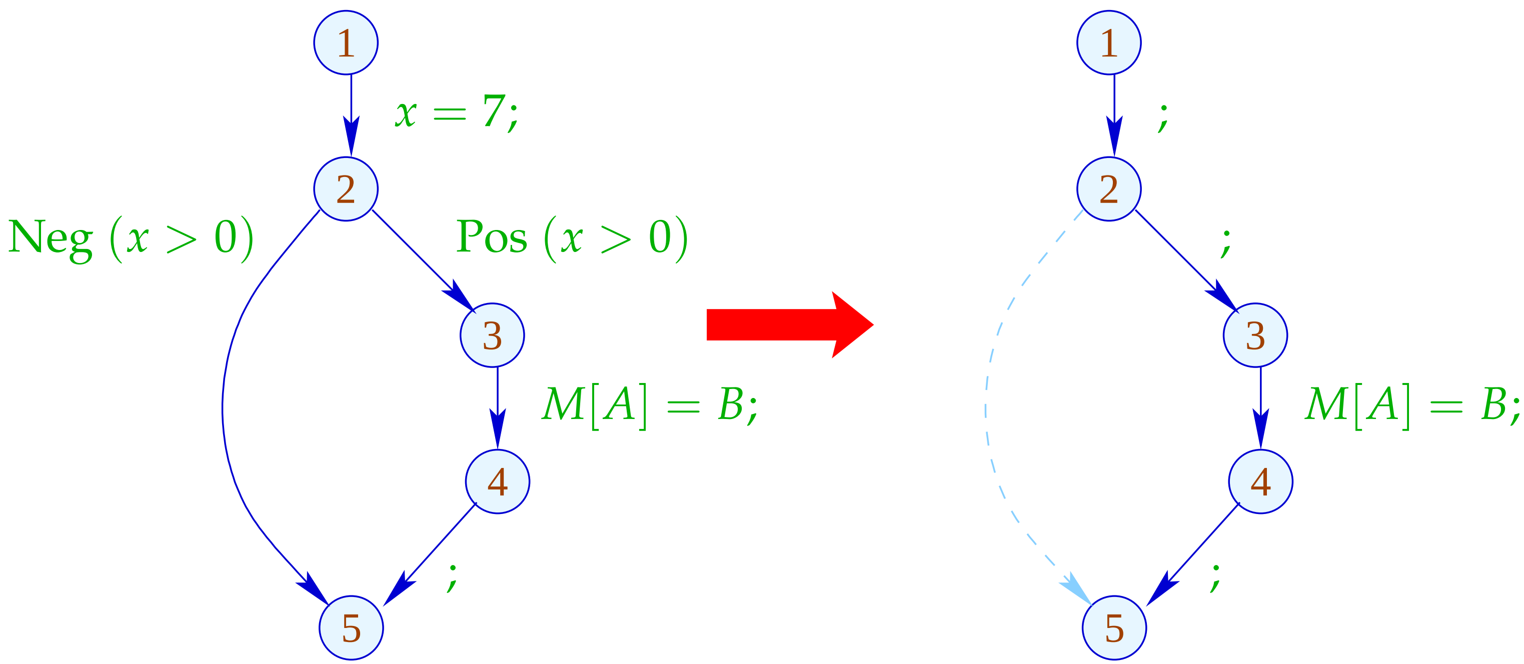
Goal:



Obviously, x has always the value 7 :-)

Thus, the memory access is always executed :-))

Goal:



Generalization: Partial Evaluation

Neil D. Jones, DIKU, Kopenhagen