Script generated by TTT

Title: Seidl: Programmoptimierung (11.11.2015)

Date: Wed Nov 11 10:10:54 CET 2015

Duration: 92:29 min

Pages: 61

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

10 : 15

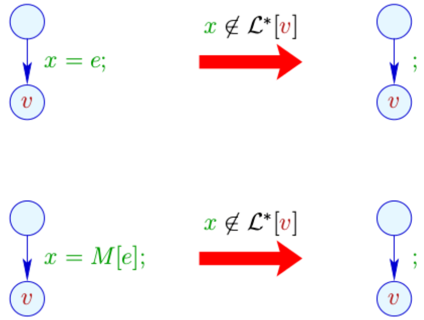
(1) Collecting constraints:

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

(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 :-))

Transformation 2:



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

(1) Collecting constraints:

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

(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 :-))

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

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

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

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:

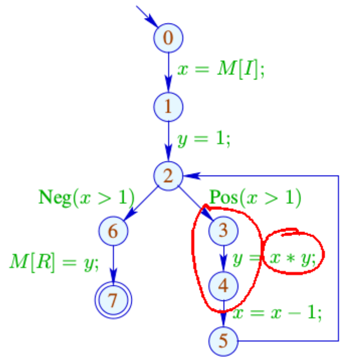
$$
\begin{array}{lll}
\mathcal{L}[stop] & \supseteq & X \\
\mathcal{L}[u] & \supseteq & [\![k]\!]^\sharp\,(\mathcal{L}[v]) \qquad k = (u, \_, v) \quad \text{edge}
\end{array}
$$

(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 :-))

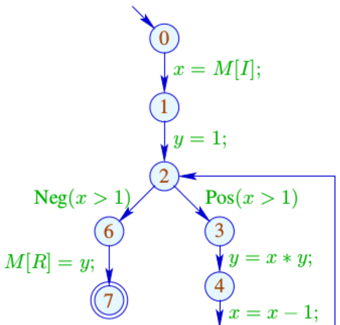
Caveat: The information is propagated backwards !!!

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:

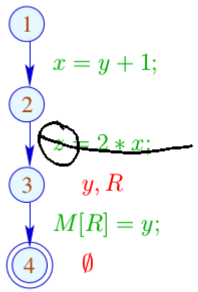


	1	2
7	$\emptyset$	
6	$\{y, R\}$	
2	$\{x, y, R\}$	ditto
5	$\{x, y, R\}$	
4	$\{x, y, R\}$	
3	$\{x, y, R\}$	
1	$\{x, R\}$	
0	$\{I, R\}$	

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

Caveat:

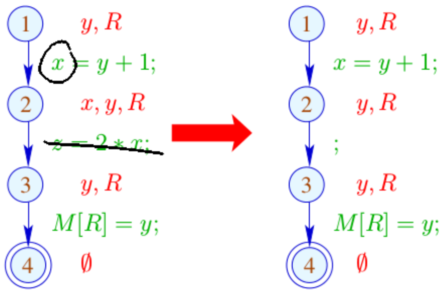
Removal of assignments to dead variables may kill further variables:



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

Caveat:

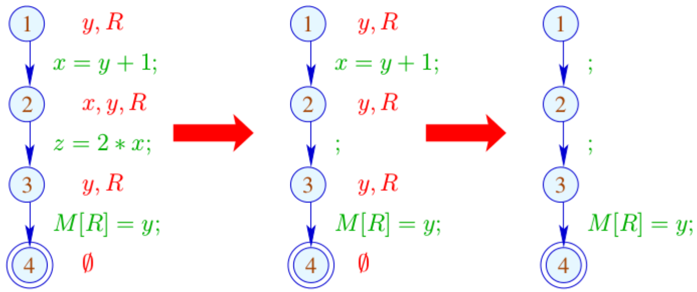
Removal of assignments to dead variables may kill further variables:



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

Caveat:

Removal of assignments to dead variables may kill further variables:



Re-analyzing the program is inconvenient :-(

Idea: Analyze true liveness!

x is called truly 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 relative to π_2;
- π_1 does not contain any definition of x.

Re-analyzing the program is inconvenient :-(

Idea: Analyze true liveness!

x is called truly 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:

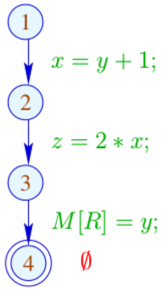
- k is a true use of x relative to π_2;
- π_1 does not contain any definition of x.

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

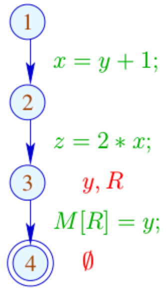
lab	truely used
;	$\emptyset$
$\text{Pos}\,(e)$	$\mathit{Vars}\,(e)$
$\text{Neg}\,(e)$	$\mathit{Vars}\,(e)$
$x = e;$	$\mathit{Vars}\,(e)$ $(*)$
$x = M[e];$	$\mathit{Vars}\,(e)$ $(*)$
$M[e_1] = e_2;$	$\mathit{Vars}(e_1) \cup \mathit{Vars}(e_2)$

$(*)$ – given that x is truely live at v w.r.t. π_2 :-)

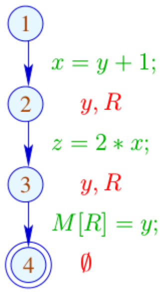
Example:



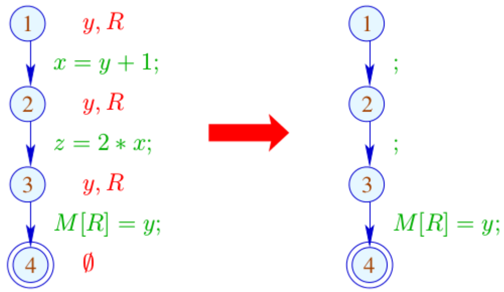
Example:



Example:



Example:



The Effects of Edges:

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

The Effects of Edges:

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

Note:

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

The Effects of Edges:

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

Note:

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

  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:

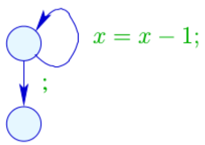
- The effects of edges for truely live variables are more complicated than for live variables :-)
- Nonetheless, they are distributive !!

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

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

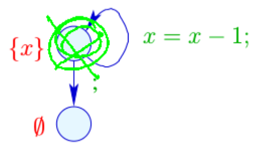
$\Longrightarrow$ the constraint system yields the MOP :-))

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



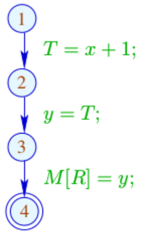
- True liveness detects more superfluous assignments than repeated liveness !!!

Liveness:



1.3 Removing Superfluous Moves

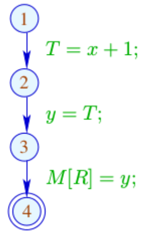
Example:



This variable-variable assignment is obviously useless :-(

1.3 Removing Superfluous Moves

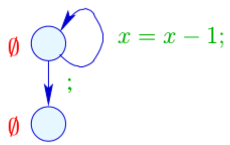
Example:



This variable-variable assignment is obviously useless :-(

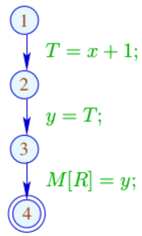
- True liveness detects more superfluous assignments than repeated liveness !!!

True Liveness:



1.3 Removing Superfluous Moves

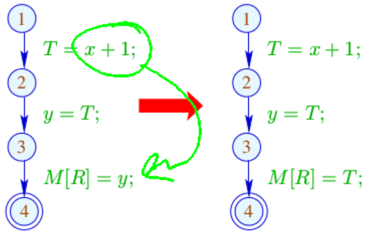
Example:



This variable-variable assignment is obviously useless :-(

1.3 Removing Superfluous Moves

Example:

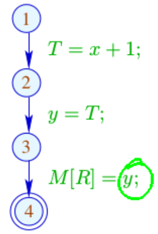


This variable-variable assignment is obviously useless :-(

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

1.3 Removing Superfluous Moves

Example:

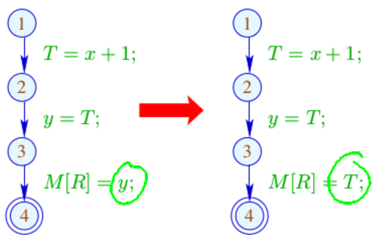


This variable-variable assignment is obviously useless :-(

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

1.3 Removing Superfluous Moves

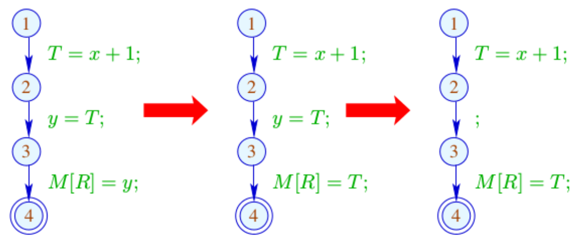
Example:



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

1.3 Removing Superfluous Moves

Example:



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

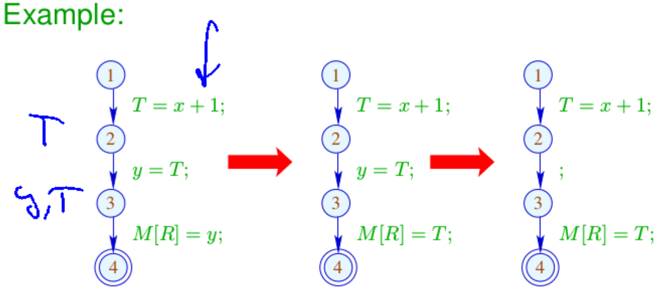
Idea:

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

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

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: $\mathbb{V} = (Expr \setminus Vars) \to 2^{Vars}$ ...

Idea:

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

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

$$[\![;]\!]^\sharp\, V = V$$

$$[\![\text{Pos}(e)]\!]^\sharp\, V\, e' = [\![\text{Neg}(e)]\!]^\sharp\, V\, e' = \begin{cases} \emptyset & \text{if } e' = e \\ V\, e' & \text{otherwise} \end{cases}$$

...

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

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

$$[\![x = e;]\!]^\sharp\, V\, e' = \begin{cases} \{x\} & \text{if } e' = e \\ (V\,e')\backslash\{x\} & \text{otherwise} \end{cases}$$

$$[\![x = M[c];]\!]^\sharp\, V\, e' = (V\,e')\backslash\{x\}$$

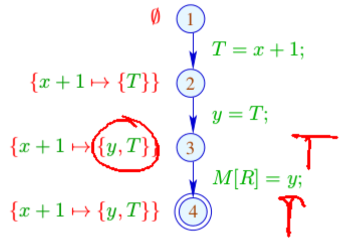
$$[\![x = M[y];]\!]^\sharp\, V\, e' = (V\,e')\backslash\{x\}$$

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

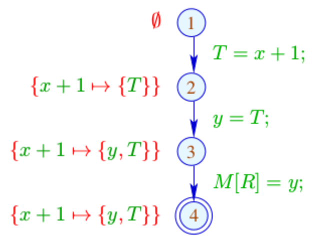
// analogously for the diverse stores

$M[e_1] = e_2$

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:

$V_1 \sqsubseteq V_2$ iff $V_1\, e \supseteq V_2\, e$ for all e

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

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

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

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

$Exp \setminus Var \to 2^{Var}$

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 :-))

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

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

$$[\![x = e;]\!]^\sharp\, V\, e' = \begin{cases} \{x\} & \text{if} \quad e' = e \\ (V\, e') \backslash \{x\} & \text{otherwise} \end{cases}$$

$$[\![x = M[c];]\!]^\sharp\, V\, e' = (V\, e') \backslash \{x\}$$

$$[\![x = M[y];]\!]^\sharp\, V\, e' = (V\, e') \backslash \{x\}$$

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

// analogously for the diverse stores

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 :-))

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

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

$$[\![x = e;]\!]^\sharp \, V \, e' = \begin{cases} \{x\} & \text{if } e' = e \\ (V \, e') \backslash \{x\} & \text{otherwise} \end{cases}$$

$$[\![x = M[c];]\!]^\sharp \, V \, e' = (V \, e') \backslash \{x\}$$

$$[\![x = M[y];]\!]^\sharp \, V \, e' = (V \, e') \backslash \{x\}$$

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

// analogously for the diverse stores

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

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

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

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

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

(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:

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

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

$\Longrightarrow$

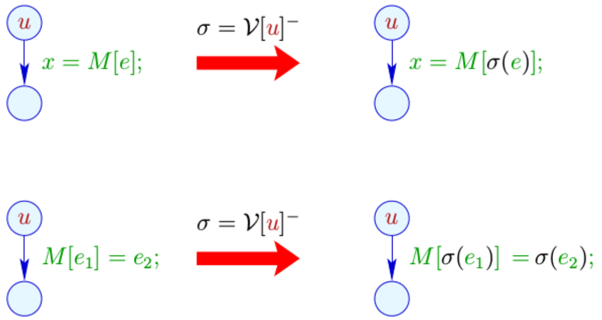
the access to x can be replaced by the access to T_e :-)

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 (cont.):



Procedure as a whole:

(1) Availability of expressions: T1

- \+ removes arithmetic operations
- − inserts superfluous moves

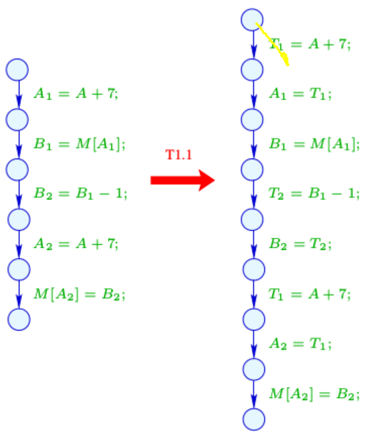
(2) Values of variables: T3

- \+ creates dead variables

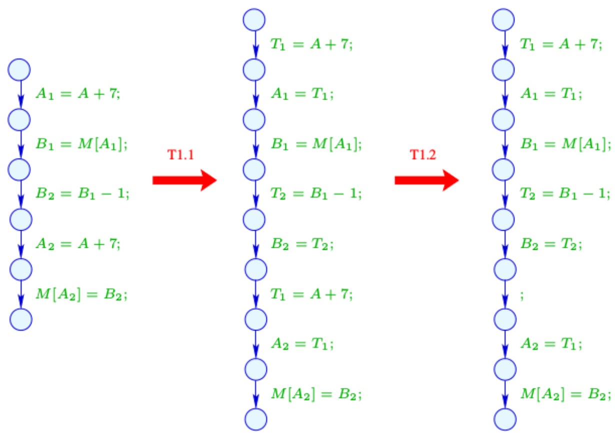
(3) (true) liveness of variables: T2

- \+ removes assignments to dead variables

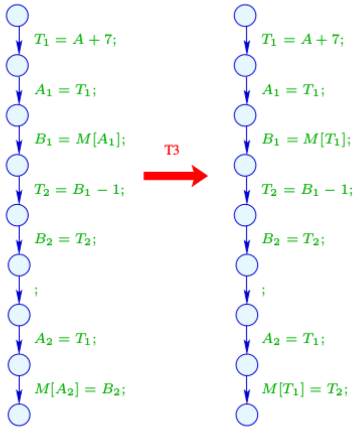
Example: `a[7]--;`



Example: `a[7]--;`



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



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