Script generated by TTT

Title: Seidl: Programmoptimierung (11.11.2013)

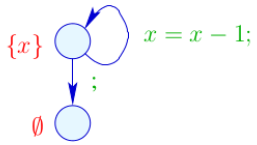
Date: Mon Nov 11 14:01:29 CET 2013

Duration: 88:18 min

Pages: 37

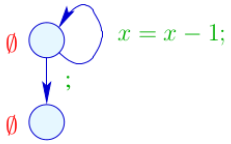
- True liveness detects more superfluous assignments than repeated liveness !!!

Liveness:



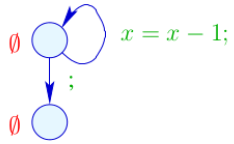
- True liveness detects more superfluous assignments than repeated liveness !!!

True Liveness:



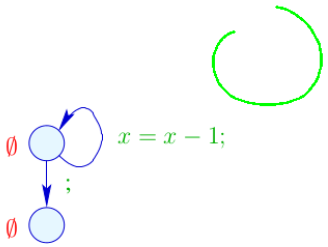
- True liveness detects more superfluous assignments than repeated liveness !!!

True Liveness:



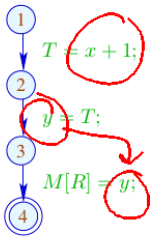
- True liveness detects more superfluous assignments than repeated liveness !!!

True Liveness:



1.3 Removing Superfluous Moves

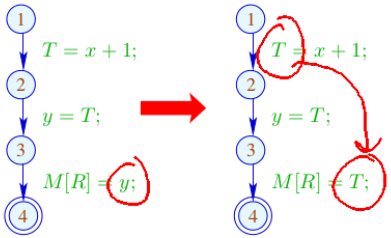
Example:



This variable-variable assignment is obviously useless :-(

1.3 Removing Superfluous Moves

Example:

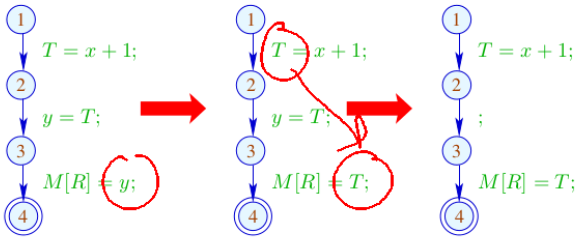


This variable-variable assignment is obviously useless :-(

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

1.3 Removing Superfluous Moves

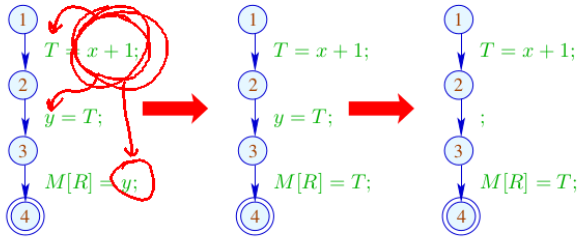
Example:



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

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:

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

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

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

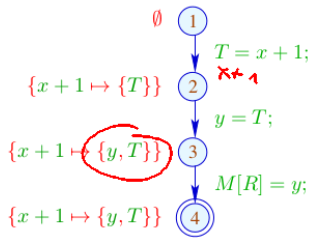
/ analogously for the diverse stores

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

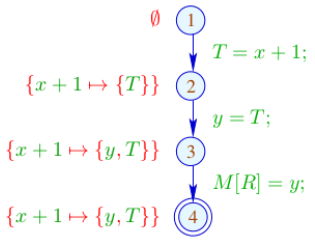
/ analogously for the diverse stores

247

In the Example:



In the Example:



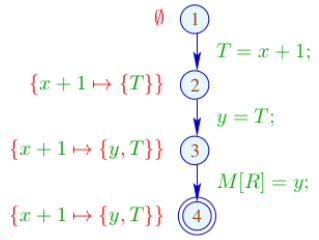
→ We propagate information in forward direction :-)

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

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

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

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:

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

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:

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

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

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:

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

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

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

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

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

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

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

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

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

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

$\Longrightarrow$

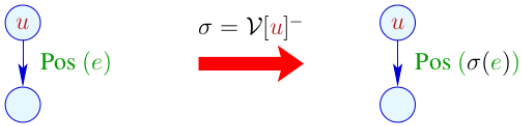
the access to x can be replaced by the access to T_e :-)

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

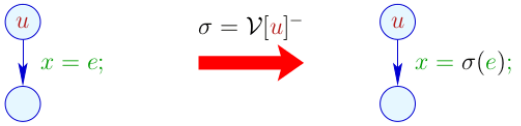
$$V^-\,x = \begin{cases} T_e & \text{if } x \in V\,e \\ x & \text{otherwise} \end{cases}$$

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

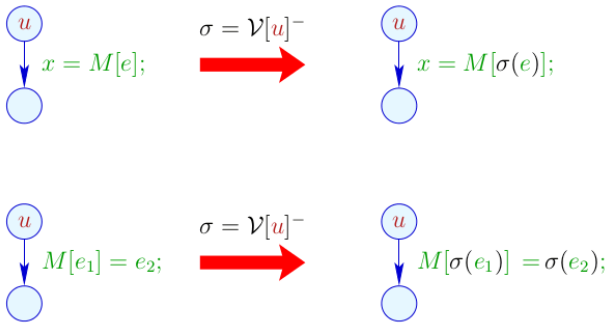
Transformation 3:



... analogously for edges with Neg (e)



Transformation 3 (cont.):



Procedure as a whole:

(1) Availability of expressions: T1

- \+ removes arithmetic operations
- – inserts superfluous moves

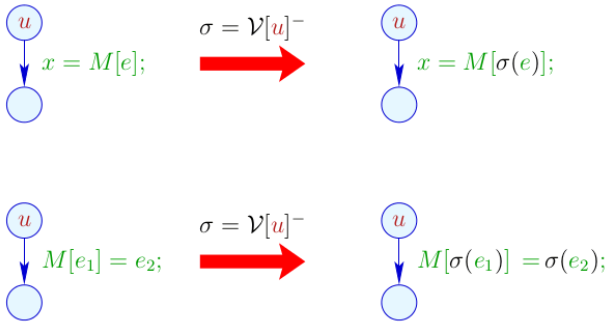
(2) Values of variables: T3

- \+ creates dead variables

(3) (true) liveness of variables: T2

- \+ removes assignments to dead variables

Transformation 3 (cont.):



256

Procedure as a whole:

(1) Availability of expressions: T1

- \+ removes arithmetic operations
- – inserts superfluous moves

(2) Values of variables: T3

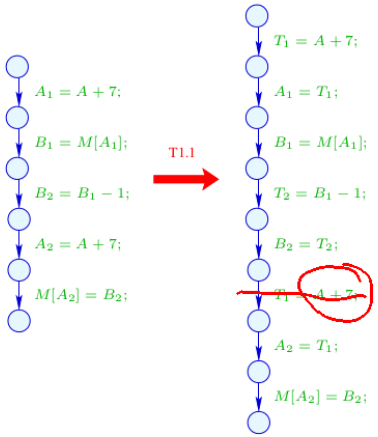
- \+ creates dead variables

(3) (true) liveness of variables: T2

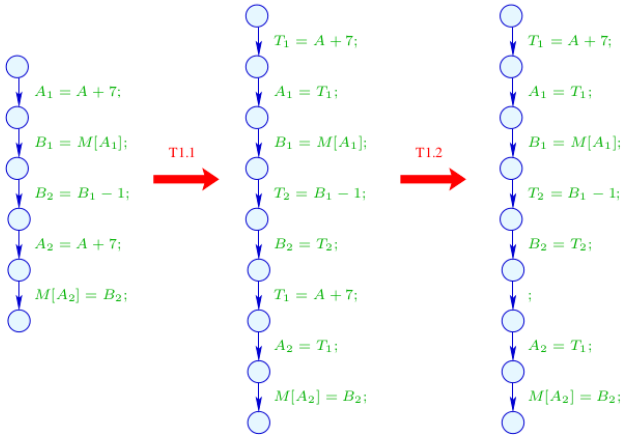
- \+ removes assignments to dead variables

257

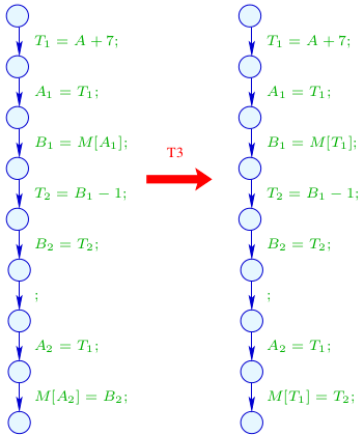
Example: `a[7]--;`



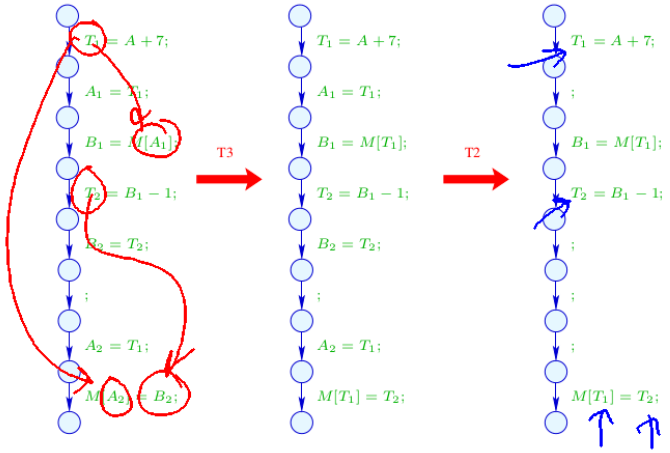
Example: `a[7]--;`



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



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



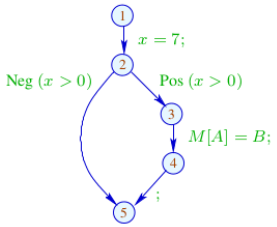
1.4 Constant Propagation

Idea:

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

Example:

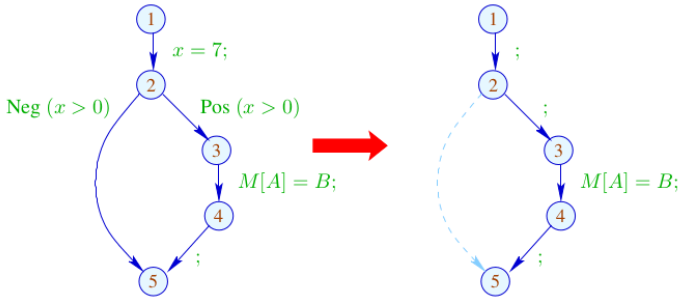
$x = 7;$
if $(x > 0)$
  $M[A] = B;$



Obviously, x has always the value 7 :-)

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

Goal:



Generalization: Partial Evaluation

Neil D. Jones, DIKU, Kopenhagen

Idea:

Design an analysis which for every u,

- determines the values which variables definitely have;
- tells whether u can be reached at all :-)

Idea:

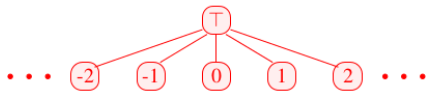
Design an analysis which for every u,

- determines the values which variables definitely have;
- tells whether u can be reached at all :-)

The complete lattice is constructed in two steps.

(1) The potential values of variables:

$$\mathbb{Z}^\top = \mathbb{Z} \cup \{\top\} \qquad \text{with} \quad x \sqsubseteq y \quad \text{iff } y = \top \text{ or } x = y$$



Caveat: $\mathbb{Z}^\top$ is not a complete lattice in itself :-(

(2) $\mathbb{D} = (\mathit{Vars} \to \mathbb{Z}^\top)_\bot = (\mathit{Vars} \to \mathbb{Z}^\top) \cup \{\bot\}$

// $\bot$ denotes: "not reachable" :-))

with $D_1 \sqsubseteq D_2$ iff $\bot = D_1$ or

$D_1\, x \sqsubseteq D_2\, x \quad (x \in \mathit{Vars})$

Remark: $\mathbb{D}$ is a complete lattice :-)

$\{x \mapsto 5,\ y \mapsto \top\}$