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 $k = (\_ , \textit{lab} , \_ )$ is defined by:

<table>
<thead>
<tr>
<th>lab</th>
<th>used</th>
<th>defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>$_ ; _ _ _ ;$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$\textit{Pos} (e)$</td>
<td>$\text{Vars} (e)$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$\textit{Neg} (e)$</td>
<td>$\text{Vars} (e)$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$x = e;$</td>
<td>$\text{Vars} (e)$</td>
<td>${x}$</td>
</tr>
<tr>
<td>$x = M[e]$;</td>
<td>$\text{Vars} (e)$</td>
<td>${x}$</td>
</tr>
<tr>
<td>$M[e_1] = e_2;$</td>
<td>$\text{Vars} (e_1) \cup \text{Vars} (e_2)$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
A variable $x$ which is not live at $u$ along $\pi$ (relative to $X$) is called \textbf{dead} at $u$ along $\pi$ (relative to $X$).

\textbf{Example:}

\begin{align*}
x &= y + 2; \quad y = 5; \quad x = y + 3;
\end{align*}

where $X = \emptyset$. Then we observe:

<table>
<thead>
<tr>
<th>$u$</th>
<th>live</th>
<th>dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{y}</td>
<td>{x}</td>
</tr>
<tr>
<td>1</td>
<td>\emptyset</td>
<td>{x, y}</td>
</tr>
<tr>
<td>2</td>
<td>{y}</td>
<td>{x}</td>
</tr>
<tr>
<td>3</td>
<td>\emptyset</td>
<td>{x, y}</td>
</tr>
</tbody>
</table>
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 $\mathbb{L} = 2^{\text{Vars}}$.

For $k = (\_ , \text{lab} , \_ )$, define $[k]^\# = [\text{lab}]^\#$ by:

\[
\begin{align*}
[;] & \vdash L & = & L \\
[\text{Pos}(e)] & \vdash L & = & [\text{Neg}(e)] L = L \cup \text{Vars}(e) \\
[x = e;] & \vdash L & = & (L \setminus \{x\}) \cup \text{Vars}(e) \\
[x = M[e];] & \vdash L & = & (L \setminus \{x\}) \cup \text{Vars}(e) \\
[M[e_1] = e_2;] & \vdash L & = & L \cup \text{Vars}(e_1) \cup \text{Vars}(e_2)
\end{align*}
\]
Let $L = 2^{\text{Vars}}$.

For $k = (\_, \text{lab}, \_)$, define $[k]^\# = [\text{lab}]^\#$ by:

$[\_;] L = L$

$[\text{Pos}(e)]^\# L = [\text{Neg}(e)]^\# L = L \cup \text{Vars}(e)$

$[x = e;] L = (L \setminus \{x\}) \cup \text{Vars}(e)$

$[x = M[e];] L = (L \setminus \{x\}) \cup \text{Vars}(e)$

$[M[e_1] = e_2;] L = L \cup \text{Vars}(e_1) \cup \text{Vars}(e_2)$

$[k]^\#$ can again be composed to the effects of $[\pi]^\#$ of paths $\pi = k_1 \ldots k_r$ by:

$[\pi]^\# = [k_1]^\# \circ \ldots \circ [k_r]^\#$
We verify that these definitions are meaningful

\[ x = y + 2; \quad y = 5; \quad x = y + 2; \quad M[y] = x; \]
We verify that these definitions are meaningful

\[ x = y + 2; \quad y = 5; \quad x = y + 2; \quad M[y] = x; \]
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

\[ x = y + 2; \quad y = 5; \quad x = y + 2; \quad M[y] = x; \]

1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5

\{y\} \quad \emptyset \quad \{y\} \quad \{x, y\} \quad \emptyset
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):

\[ x = e; \]

\[ x \notin \mathcal{L}^*[v] \]

\[ x = M[e]; \]

\[ x \notin \mathcal{L}^*[v] \]
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 \([\pi]^\sharp\ L\) 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:

$$\mathcal{L}[\text{stop}] \supseteq X$$

$$\mathcal{L}[u] \supseteq [[k]]^\# (\mathcal{L}[v]) \quad k = (u, _, v) \quad \text{edge}$$

(2) Solving the constraint system by means of RR iteration. Since $\mathcal{L}$ 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 $[[k]]^\#$ are distributive.
Computation of the sets $\mathcal{L}^*[u]$:

(1) Collecting constraints:

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

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

Since $\mathcal{L}$ 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:

\[ x = M[I]; \]
\[ y = 1; \]
\[ M[R] = y; \]
\[ y = x \ast y; \]
\[ x = x - 1; \]

\[ \mathcal{L}[0] \supseteq (\mathcal{L}[1]\{x\}) \cup \{I\} \]
\[ \mathcal{L}[1] \supseteq \mathcal{L}[2]\{y\} \]
\[ \mathcal{L}[2] \supseteq (\mathcal{L}[6] \cup \{x\}) \cup (\mathcal{L}[3] \cup \{x\}) \]
\[ \mathcal{L}[3] \supseteq (\mathcal{L}[4]\{y\}) \cup \{x, y\} \]
\[ \mathcal{L}[4] \supseteq (\mathcal{L}[5]\{x\}) \cup \{x\} \]
\[ \mathcal{L}[5] \supseteq \mathcal{L}[2] \]
\[ \mathcal{L}[6] \supseteq \mathcal{L}[7] \cup \{y, R\} \]
\[ \mathcal{L}[7] \supseteq \emptyset \]
Example:

\[ x = M[I]; \]

\[ y = 1; \]

\[ M[R] = y; \]

\[ y = x \ast y; \]

\[ x = x - 1; \]

\[ \text{Neg}(x > 1) \]

\[ \text{Pos}(x > 1) \]

\[
\begin{array}{c|c|c}
\text{1} & \text{2} \\
7 & \emptyset & \text{dito} \\
6 & \{y, R\} & \\
5 & \{x, y, R\} & \\
4 & \{x, y, R\} & \\
3 & \{x, y, R\} & \\
2 & \{x, y, R\} & \\
1 & \{x, R\} & \\
0 & \{I, R\} & \\
\end{array}
\]
The left-hand side of no assignment is dead

Caveat:

Removal of assignments to dead variables may kill further variables:

\[ x = y + 1; \]
\[ z = 2 \times x; \]
\[ M[R] = y; \]
\[ \emptyset \]
The left-hand side of no assignment is dead

Caveat:

Removal of assignments to dead variables may kill further variables:

1. $x = y + 1$
2. $z = 2 \times x$
3. $M[R] = y$
4. $\emptyset$
The left-hand side of no assignment is **dead**

**Caveat:**

Removal of assignments to dead variables may kill further variables:

\[
1. \quad x = y + 1; \\
2. \quad x, y, R \\
3. \quad z = 2 \times x; \\
4. \quad y, R \\
5. \quad M[R] = y; \\
6. \quad \emptyset
\]
The left-hand side of no assignment is **dead**

**Caveat:**

Removal of assignments to dead variables may kill further variables:

1. $y, R$
   
   $x = y + 1$;

2. $x, y, R$
   
   $z = 2 * x$;

3. $y, R$
   
   $M[R] = y$;

4. $\emptyset$
The left-hand side of no assignment is **dead**

**Caveat:**

Removal of assignments to dead variables may kill further variables:

1. $y, R$
   
   $x = y + 1$;

2. $x, y, R$
   
   $z = 2 \times x$;

3. $y, R$
   
   $M[R] = y$;

4. $\emptyset$

![Diagram showing the removal of assignments to dead variables](diagram.png)
The left-hand side of no assignment is **dead**

**Caveat:**

Removal of assignments to dead variables may kill further variables:

1. $y, R$
2. $x = y + 1$;
3. $x, y, R$
4. $z = 2 \times x$;
5. $y, R$
6. $M[R] = y$;
7. $\emptyset$
8. $y, R$
The left-hand side of no assignment is **dead**

**Caveat:**

Removal of assignments to dead variables may kill further variables:

\[
\begin{align*}
1 & \quad y, R \\
2 & \quad x = y + 1; \\
3 & \quad x, y, R \\
4 & \quad z = 2 \times x; \\
\end{align*}
\]

\[
\begin{align*}
1 & \quad y, R \\
2 & \quad x = y + 1; \\
3 & \quad y, R \\
4 & \quad M[R] = y; \\
\end{align*}
\]

\[
\begin{align*}
1 & \quad y, R \\
2 & \quad x = y + 1; \\
3 & \quad y, R \\
4 & \quad M[R] = y; \\
\end{align*}
\]
Re-analyzing the program is inconvenient

Idea: Analyze true liveness!

\( x \) is called \textbf{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 \textbf{true} use of \( x \);
- \( \pi_1 \) does not contain any \textbf{definition} of \( x \).
The set of truly used variables at an edge $k = (\_, \text{lab}, \nu)$ is defined as:

<table>
<thead>
<tr>
<th>lab</th>
<th>truly used</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$\text{Pos}(e)$</td>
<td>$\text{Vars}(e)$</td>
</tr>
<tr>
<td>$\text{Neg}(e)$</td>
<td>$\text{Vars}(e)$</td>
</tr>
<tr>
<td>$x = e;$</td>
<td>$\text{Vars}(e)$ ($\ast$)</td>
</tr>
<tr>
<td>$x = M[e];$</td>
<td>$\text{Vars}(e)$ ($\ast$)</td>
</tr>
<tr>
<td>$M[e_1] = e_2;$</td>
<td>$\text{Vars}(e_1) \cup \text{Vars}(e_2)$</td>
</tr>
</tbody>
</table>

($\ast$) – given that $x$ is truly live at $\nu$
Example:

\[ x = y + 1; \]
\[ z = 2 \times x; \]
\[ M[R] = y; \]
\[ \emptyset \]
Example:

1. \( x = y + 1; \)
2. \( z = 2 \times x; \)
3. \( y, R \)
   \[ M[R] = y; \]
4. \( \emptyset \)
Example:

\[ x = y + 1; \]
\[ y, R \]
\[ z = 2 \times x; \]
\[ y, R \]
\[ M[R] = y; \]
\[ \emptyset \]
Example:

1. $y, R$
   
   $x = y + 1$;

2. $y, R$
   
   $z = 2 \times x$;

3. $y, R$
   
   $M[R] = y$;

4. $\emptyset$
Example:

\[ x = y + 1; \]
\[ z = 2 \times x; \]
\[ M[R] = y; \]
\[ \emptyset \]
The Effects of Edges:

\[
\begin{align*}
[;]^{\#} L &= L \\
[\text{Pos}(e)]^{\#} L &= [\text{Neg}(e)]^{\#} L = L \cup \text{Vars}(e) \\
[x = e;]^{\#} L &= (L \setminus \{x\}) \cup \text{Vars}(e) \\
[x = M[e];]^{\#} L &= (L \setminus \{x\}) \cup \text{Vars}(e) \\
[M[e_1] = e_2;]^{\#} L &= L \cup \text{Vars}(e_1) \cup \text{Vars}(e_2)
\end{align*}
\]
The Effects of Edges:

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

$$f (y_1 \cup y_2) = (u \in y_1 \cup y_2) ? b : \emptyset$$
$$= (u \in y_1 \lor u \in y_2) ? b : \emptyset$$
$$= (u \in y_1) ? b : \emptyset \cup (u \in y_2) ? b : \emptyset$$
$$= f y_1 \cup f y_2$$
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:

\[
f(y_1 \cup y_2) = (u \in y_1 \cup y_2) ? b : \emptyset \\
= (u \in y_1 \lor u \in y_2) ? b : \emptyset \\
= (u \in y_1) ? b : \emptyset \cup (u \in y_2) ? b : \emptyset \\
= f y_1 \cup f y_2
\]

\( \implies \) the constraint system yields the MOP
• True liveness detects more superfluous assignments than repeated liveness !!!

\[ x = x - 1; \]
True liveness detects more superfluous assignments than repeated liveness !!!

Liveness:

\[
\{x\} \xrightarrow{x = x - 1;} \emptyset
\]
• True liveness detects more superfluous assignments than repeated liveness !!!

True Liveness:

\[
x = x - 1;
\]
1.3 Removing Superfluous Moves

Example:

\[
\begin{align*}
T &= x + 1; \\
y &= T; \\
M[R] &= y;
\end{align*}
\]

This variable-variable assignment is obviously useless
1.3 Removing Superfluous Moves

Example:

1. $T = x + 1;
2. y = T;
3. $M[R] = y;
4. $M[R] = y;

This variable-variable assignment is obviously useless
Instead of $y$, we could also store $T$
1.3 Removing Superfluous Moves

Example:

\[ T = x + 1; \]
\[ y = T; \]
\[ M[R] = y; \]

This variable-variable assignment is obviously useless.
Instead of \( y \), we could also store \( T \).
1.3 Removing Superfluous Moves

Example:

```
T = x + 1;
y = T;
M[R] = y;
```

Advantage: Now, \( y \) has become \textit{dead}
1.3 Removing Superfluous Moves

Example:

\[ T = x + 1; \]
\[ y = T; \]
\[ M[R] = y; \]

Advantage: Now, \( y \) has become dead
Idea:

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

We use: \( \nabla = Expr \rightarrow 2^{Vars} \) ...
Idea:

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

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

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

// analogously for the diverse stores
In the Example:

\[ T = x + 1; \]
\[ y = T; \]
\[ M[R] = y; \]
In the Example:

We propagate information in \textit{forward} direction

At \textit{start}, \( V_0 e = \emptyset \) for all \( e \);

\( \subseteq \subseteq \mathbb{V} \times \mathbb{V} \) is defined by:

\[
V_1 \subseteq V_2 \iff V_1 e \supseteq V_2 e \quad \text{for all} \quad e
\]
Observation:

The new effects of edges are distributive:

To show this, we consider the functions:

(1) \( f^x_V e = (V e) \setminus \{x\} \)

(2) \( f^{e,a}_V = V \oplus \{e \mapsto a\} \)

(3) \( f^{x,y}_V e = (y \in V e) ? (V e \cup \{x\}) : ((V e) \setminus \{x\}) \)

Obviously, we have:

\[
\begin{align*}
[x = e;] & = f^{e,\{x\}}_2 \circ f^x_1 \\
[x = y;] & = f^{x,y}_3 \\
[x = M[e];] & = f^{e,\emptyset}_2 \circ f^x_1
\end{align*}
\]

By closure under composition, the assertion follows
(1) For \( f V e = (V e) \setminus \{x\} \), we have:

\[
f (V_1 \sqcup V_2) e = ((V_1 \sqcup V_2) e) \setminus \{x\} \\
= ((V_1 e) \cap (V_2 e)) \setminus \{x\} \\
= ((V_1 e) \setminus \{x\}) \cap ((V_2 e) \setminus \{x\}) \\
= (f V_1 e) \cap (f V_2 e) \\
= (f V_1 \sqcup f V_2) e
\]
For $f V = V \oplus \{ e \mapsto a \}$, we have:

\[
\begin{align*}
f(\bigvee V_1 \sqcup V_2) e' &= ((\bigvee V_1 \sqcup V_2) \oplus \{ e \mapsto a \}) e' \\
&= (\bigvee V_1 \sqcup V_2) e' \\
&= (f V_1 \sqcup f V_2) e' \quad \text{given that } e \neq e' \\
f(\bigvee V_1 \sqcup V_2) e &= ((\bigvee V_1 \sqcup V_2) \oplus \{ e \mapsto a \}) e \\
&= (\bigvee V_1 \oplus \{ e \mapsto a \}) e \cap ((\bigvee V_2 \oplus \{ e \mapsto a \}) e) \\
&= (f V_1 \sqcup f V_2) e
\end{align*}
\]
(3) For \( f V e = (y \in V e) \? (V e \cup \{x\}) : (\{x\} \setminus \{x\}) \), we have:

\[
\begin{align*}
  f (V_1 \sqcup V_2) e & = (((V_1 \sqcup V_2) e) \setminus \{x\}) \cup (y \in (V_1 \sqcup V_2) e) ? \{x\} : \emptyset \\
  & = ((V_1 e \cap V_2 e) \setminus \{x\}) \cup (y \in (V_1 e \cap V_2 e)) ? \{x\} : \emptyset \\
  & = ((V_1 e \cap V_2 e) \setminus \{x\}) \cup \\
  & \quad ((y \in V_1 e) ? \{x\} : \emptyset) \cap ((y \in V_2 e) ? \{x\} : \emptyset) \\
  & = (((V_1 e) \setminus \{x\}) \cup (y \in V_1 e) ? \{x\} : \emptyset) \cap \\
  & \quad (((V_2 e) \setminus \{x\}) \cup (y \in V_2 e) ? \{x\} : \emptyset) \\
  & = (f V_1 \sqcup f V_2) e
\end{align*}
\]
We conclude:

→ Solving the constraint system returns the MOP solution

→ Let \( V \) denote this solution.

If \( x \in V[u]e \), then \( x \) at \( u \) contains the value of \( e \) — which we have stored in \( T_e \)

\[ T_e \]

⇒ 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:

\[
V^{-} x = \begin{cases} 
T_e & \text{if } x \in Ve \\
T & \text{otherwise}
\end{cases}
\]

if \( Ve \cap Ve' = \emptyset \) for \( e \neq e' \). Otherwise: \( V^{-} x = x \)
Transformation CE:

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

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

\[ x = e; \]

\[ x = \sigma(e); \]
Transformation CE (cont.):

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

\[ x = M[e]; \quad \sigma = \mathcal{V}[u]^{-} \quad x = M[\sigma(e)]; \]

\[ M[e_1] = e_2; \quad \sigma = \mathcal{V}[u]^{-} \quad M[\sigma(e_1)] = \sigma(e_2); \]
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: \( \text{a[7]}--; \)
Example:  
\( a[7]--; \)

\[\begin{align*}
A_1 &= A + 7; \\
B_1 &= M[A_1]; \\
B_2 &= B_1 - 1; \\
A_2 &= A + 7; \\
M[A_2] &= B_2;
\end{align*}\]

\[\begin{align*}
T_1 &= A + 7; \\
A_1 &= T_1; \\
B_1 &= M[A_1]; \\
T_2 &= B_1 - 1; \\
B_2 &= T_2; \\
T_1 &= A + 7; \\
A_2 &= T_1; \\
M[A_2] &= B_2;
\end{align*}\]

\[\begin{align*}
T_1 &= A + 7; \\
A_1 &= T_1; \\
B_1 &= M[A_1]; \\
T_2 &= B_1 - 1; \\
B_2 &= T_2; \\
; \\
A_2 &= T_1; \\
M[A_2] &= B_2;
\end{align*}\]
Example (cont.): 

\[ a[7]--; \]

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

\[ T_1 = A + 7; \]
\[ A_1 = T_1; \]
\[ B_1 = M[T_1]; \]
\[ T_2 = B_1 - 1; \]
\[ B_2 = T_2; \]
\[ ; \]
\[ A_2 = T_1; \]
\[ M[T_1] = T_2; \]
Example (cont.): \( a[7]--; \)

\[
T_1 = A + 7; \\
A_1 = T_1; \\
B_1 = M[A_1]; \\
T_2 = B_1 - 1; \\
B_2 = T_2; \\
\vdots \\
A_2 = T_1; \\
M[A_2] = B_2; \\
\]

\[
T_1 = A + 7; \\
A_1 = T_1; \\
B_1 = M[T_1]; \\
T_2 = B_1 - 1; \\
B_2 = T_2; \\
\vdots \\
A_2 = T_1; \\
M[T_1] = T_2; \\
\]

\[
T_1 = A + 7; \\
A_1 = T_1; \\
B_1 = M[T_1]; \\
T_2 = B_1 - 1; \\
B_2 = T_2; \\
\vdots \\
A_2 = T_1; \\
M[T_1] = T_2; \\
\]