Thereby, the set of all defined or used variables at an edge $k = (\_, lab, \_)$ 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>Pos $(e)$</td>
<td>$\text{Vars} (e)$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>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 \textit{dead} at \( u \) along \( \pi \) (relative to \( X \)).

\textbf{Example:}

\[ x = y + 2; \quad y = 5; \quad x = y + 3; \]

\[
\begin{array}{c}
0 \quad 1 \quad 2 \quad 3 \\
\end{array}
\]

where \( X = \emptyset \). Then we observe:

<table>
<thead>
<tr>
<th></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 \( L = 2^{V_{\text{ars}}} \).

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

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

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

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

\([k]^\#\) 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 :-)

\[ M[y] = x; \]

\[ x = y + 2; \quad y = 5; \quad x = y + 2; \]
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 :-)

\[
\begin{align*}
M[y] &= x; \\
x &= y + 2; \\
y &= 5; \\
x &= y + 2; \\
\emptyset \{x, y\} \{y\} \emptyset
\end{align*}
\]
We verify that these definitions are meaningful :-)}

\[
\begin{align*}
M[y] &= x; \\
x &= y + 2; \\
y &= 5; \\
x &= y + 2; \\
\end{align*}
\]
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 $\mathcal{L}$ we use $\sqsubseteq = \subseteq$.
- The set of variables which are live at program exit is given by the set $X$ :-)}
Transformation 2:

\[ x = e; \quad x \not\in \mathcal{L}^*[v] \]

\[ x = M[e]; \quad x \not\in \mathcal{L}^*[v] \]
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 $[\pi] \# L$ 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:

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

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

(1) Collecting constraints:

$$\begin{align*}
\mathcal{L}[\text{stop}] & \supseteq X \\
\mathcal{L}[u] & \supseteq \llbracket k \rrbracket^\# (\mathcal{L}[v]) & k = (u, _, v) \text{ edge}
\end{align*}$$

(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:

\[ x = M[I]; \]

\[ y = 1; \]

Neg(\(x > 1\))

Pos(\(x > 1\))

\[ M[R] = y; \]

\[ y = x \star y; \]

\[ x = x - 1; \]

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

\begin{align*}
    y & = 1; \\
    M[R] & = y; \\
    x & = M[I]; \\
    y & = x \ast y; \\
    x & = x - 1;
\end{align*}

\begin{itemize}
    \item Pos\((x > 1)\)
    \item Neg\((x > 1)\)
\end{itemize}

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>\emptyset</td>
<td>dito</td>
</tr>
<tr>
<td>6</td>
<td>{y, R}</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>{x, y, R}</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>{x, y, R}</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>{x, y, R}</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>{x, y, R}</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>{x, R}</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>{I, R}</td>
<td></td>
</tr>
</tbody>
</table>
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:

```
x = y + 1;
z = 2 * x;
y, R
M[R] = y;
∅
```
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

\[ x, y, R \]

\[ z = 2 \times x; \]

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

**Caveat:**

Removal of assignments to dead variables may kill further variables:

$$
\begin{array}{l}
1. y, R \\
x = y + 1; \\
2. x, y, R \\
z = 2 \times x; \\
3. y, R \\
M[R] = y; \\
4. \emptyset
\end{array}
\quad \rightarrow \quad
\begin{array}{l}
1. x = y + 1; \\
2. \\
3. y, R \\
M[R] = y; \\
4. \emptyset
\end{array}
$$
The left-hand side of no assignment is **dead**  :-)  

**Caveat:**

Removal of assignments to dead variables may kill further variables:

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

\[
\begin{align*}
  &1: y, R \\
  &2: x = y + 1; \\
  &3: y, R \\
  &4: M[R] = y; \\
  &\emptyset
\end{align*}
\]
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 truely 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 true use of \( x \);
- \( \pi_1 \) does not contain any definition of \( x \).
The set of truely used variables at an edge $k = (_, lab, v)$ is defined as:

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

(*) – given that $x$ is truely live at $v$ :-)

226
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:

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

1. $y, R$
2. $x = y + 1$;
3. $y, R$
4. $z = 2 \times x$;
5. $y, R$
6. $M[R] = y$;
7. $\emptyset$

$230$
Example:

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

1. \( M[R] = y; \)
2. \( \emptyset \)
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 \( D = 2^U \), \( f y = (u \in y) ? b : \emptyset \)  
We verify:

\[
\begin{align*}
    f (y_1 \cup y_2) &= (u \in y_1 \cup y_2) ? b : \emptyset \\
                    &= (u \in y_1 \lor u \in y_2) ? b : \emptyset \\
                    &= (u \in y_1) ? b : \emptyset \cup (u \in y_2) ? b : \emptyset \\
                    &= f y_1 \cup f y_2
\end{align*}
\]
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 $\mathbb{D} = 2^U$, $f y = (u \in y) \dashv b : \emptyset$ We verify:

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

$\implies$ 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:

\[
\begin{align*}
\{x\} & \quad x = x - 1; \\
\emptyset & \quad ;
\end{align*}
\]
• True liveness detects more superfluous assignments than repeated liveness !!!

True Liveness:

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

Example:

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

This variable-variable assignment is obviously useless  :-(
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:

$T = x + 1$;

$y = T$;

$M[R] = y$;

$T = x + 1$;

$y = T$;

$M[R] = T$;

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 dead :-))
1.3 Removing Superfluous Moves

Example:

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

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

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

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

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

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