Summary and Application:

→ The effects of edges of the analysis of availability of expressions are distributive:

\[(a \cup (x_1 \cap x_2)) \setminus b = ((a \cup x_1) \cap (a \cup x_2)) \setminus b \]
\[= ((a \cup x_1) \setminus b) \cap ((a \cup x_2) \setminus b)\]

→ If all effects of edges are distributive, then the MOP can be computed by means of the constraint system and RR-iteration. :-)

→ If not all effects of edges are distributive, then RR-iteration for the constraint system at least returns a safe upper bound to the MOP.

→ The effects of edges of the analysis of availability of expressions are distributive:

\[(a \cup (x_1 \cap x_2)) \setminus b = ((a \cup x_1) \cap (a \cup x_2)) \setminus b \]
\[= ((a \cup x_1) \setminus b) \cap ((a \cup x_2) \setminus b)\]
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:

\( \rightarrow \) Assignments to dead variables can be removed \(:-\)  
\( \rightarrow \) 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 \( x \) 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 \( k = (\_, \text{lab}, \_ \_ \) 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}(c)</td>
<td>\emptyset</td>
</tr>
<tr>
<td>Neg ( e )</td>
<td>\text{Vars}(c)</td>
<td>\emptyset</td>
</tr>
<tr>
<td>( x = e; )</td>
<td>\text{Vars}(c)</td>
<td>{x}</td>
</tr>
<tr>
<td>( x = M[e]; )</td>
<td>\text{Vars}(c)</td>
<td>{x}</td>
</tr>
<tr>
<td>( M[c_1] = c_2; )</td>
<td>\text{Vars}(c_1) \cup \text{Vars}(c_2)</td>
<td>\emptyset</td>
</tr>
</tbody>
</table>

A variable \( x \) which is not live at \( u \) along \( \pi \) (relative to \( X \)) is called **dead** at \( u \) along \( \pi \) (relative to \( X \)).

**Example:**

\[
\begin{array}{c}
\text{x = y + 2;} \\
y = 5; \\
x = y + 3;
\end{array}
\]

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 \( X \)).

**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 \( [k]^F \) which transforms the set of variables which are live at \( v \) into the set of variables which are live at \( u \).

\[
\begin{align*}
[\ ]^FL & = L \\
[\operatorname{Pos}(e)]^FL & = [\operatorname{Neg}(e)]^FL = L \cup \operatorname{Vars}(e) \\
[x = c]^FL & = (L \setminus \{x\}) \cup \operatorname{Vars}(c) \\
[x = M[e]]^FL & = (L \setminus \{x\}) \cup \operatorname{Vars}(c) \\
[M[e] = e_2]^FL & = L \cup \operatorname{Vars}(e_1) \cup \operatorname{Vars}(e_2)
\end{align*}
\]

Let \( L = 2^{\operatorname{Vars}} \).

For \( k = (_-, lab, _) \), define \( [k]^F = [lab]^F \) by:

\[
\begin{align*}
[\ ]^FL & = L \\
[\operatorname{Pos}(e)]^FL & = [\operatorname{Neg}(e)]^FL = L \cup \operatorname{Vars}(e) \\
[x = c]^FL & = (L \setminus \{x\}) \cup \operatorname{Vars}(c) \\
[x = M[e]]^FL & = (L \setminus \{x\}) \cup \operatorname{Vars}(c) \\
[M[e] = e_2]^FL & = L \cup \operatorname{Vars}(e_1) \cup \operatorname{Vars}(e_2)
\end{align*}
\]

\([k]^F\) can again be composed to the effects of \( [x]^F \) of paths \( \pi = k_1 \ldots k_r \) by:

\[
[x]^F = [k_1]^F \circ \ldots \circ [k_r]^F
\]

We verify that these definitions are meaningful :)
We verify that these definitions are meaningful :)

```
1  2  3  4  5
\{y\}  0  \{x,y\}  \emptyset
```

The set of variables which are live at \( u \) then is given by:

\[
\mathcal{L}^*[u] = \bigcup \{[x] \mid X : u \rightarrow^* \text{stop} \}
\]

... literally:
- The paths start in \( u \) :-)
  \[\Rightarrow\] As partial ordering for \( L \) we use \( \subseteq \subseteq \).
- The set of variables which are live at program exit is given by the set \( X \) :-)

Transformation 2:
```
\begin{align*}
& x = e; \\
& x \notin \mathcal{L}^*[u] \\
& x = M[e]; \\
& x \notin \mathcal{L}^*[u]
\end{align*}
```

\[\leftarrow\]
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 \( \pi^2 L \) is the set of variables which are live at the beginning of \( \pi \) \( \vdash \)

→ 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 \( \vdash \)

→ Correctness of the transformation: In any execution of the transformed programs, the live variables always receive the same values \( \vdash \)

Transformation 2:

\[ x = \phi_i \]

\[ x \notin L^*[v] \]

\[ x = M[e]; \]

\[ x \notin L^*[v] \]

Computation of the sets \( L^*[u] \):

1. Collecting constraints:

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

\[ L[u] \supseteq \{ k \} (L[v]) \]

\[ k = (u, v) \text{ edge} \]

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

Since \( L \) is finite, the iteration will terminate \( \vdash \)

3. If the exit is (formally) reachable from every program point, then the smallest solution \( L \) of the constraint system equals \( L^* \) since all \( \{ k \} \) are distributive \( \vdash \)
**Transformation 2:**

1. $x = e;\quad x \notin \mathcal{L}^*[v]$  
2. $x = M[e];\quad x \notin \mathcal{L}^*[v]$  

---

We verify that these definitions are meaningful :)  

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

---

**Example:**

- $L[0] \supseteq (L[1]\{x\}) \cup \{I\}$
- $L[1] \supseteq L[2]\{y\}$
- $L[2] \supseteq (L[3]\{x\}) \cup \{L[4] \cup \{x\}\}$
- $L[3] \supseteq (L[4]\{y\}) \cup \{x\}$
- $L[4] \supseteq L[5]\{x\}$
- $L[5] \supseteq \emptyset$

---

**Example:**

\[ \begin{array}{c|c|c}
    \text{1} & \text{2} \\
    \hline
    7 & 0 \\
    6 & \{y, R\} \\
    5 & \{x, y, R\} \\
    4 & \{x, y, R\} \\
    3 & \{x, y, R\} \\
    2 & \{x, y, R\} \\
    1 & \{x, R\} \\
    0 & \{I, R\} \\
\end{array} \]
Example:

\[ z = M[R]; \]
\[ y = 1; \]
\[ \text{Neg}(x > 1) \]
\[ \text{Pos}(x > 1) \]
\[ M[R] = y; \]

---

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

---

Caveat:

The left-hand side of no assignment is \textbf{dead} \ :-) 

Removal of assignments to dead variables may kill further variables:

---

Caveat:

The left-hand side of no assignment is \textbf{dead} \ :-) 

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:

\[
\begin{array}{c}
\begin{array}{c}
1 \\
2 \\
3 \\
4 \\
\end{array}
\end{array}
\begin{array}{c}
y, R \\
x = y + 1; \\
z = 2 * x; \\
M[R] = y; \\
\emptyset \\
\end{array}
\begin{array}{c}
1 \\
2 \\
3 \\
4 \\
\end{array}
\begin{array}{c}
y, R \\
x = y + 1; \\
y, R \\
M[R] = y; \\
\emptyset \\
\end{array}
\end{array}
\]

Re-analyzing the program is inconvenient :-(

Idea: Analyze true liveness!

\( x \) is called **true** 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 \( \pi = \pi_1 \cdot \pi_2 \) such that:

- \( \pi_1 \) does not contain any definition of \( x \).
- \( x \) is a true use of \( \pi \).

The set of truly used variables at an edge \( k = (u, lab, v) \) 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>Pos(( e ))</td>
<td>Vars(( e ))</td>
</tr>
<tr>
<td>Neg(( e ))</td>
<td>Vars(( e ))</td>
</tr>
<tr>
<td>( x = e );</td>
<td>Vars(( e )) (( * ))</td>
</tr>
<tr>
<td>( x = M[e] );</td>
<td>Vars(( e )) (( * ))</td>
</tr>
<tr>
<td>( M[e_1] = e_2 );</td>
<td>Vars(( e_1 )) \cup Vars(( e_2 ))</td>
</tr>
</tbody>
</table>

(\( * \)) - given that \( x \) is truly live \( \Rightarrow \)

The set of truly used variables at an edge \( k = (u, lab, v) \) 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>Pos(( e ))</td>
<td>Vars(( e ))</td>
</tr>
<tr>
<td>Neg(( e ))</td>
<td>Vars(( e ))</td>
</tr>
<tr>
<td>( x = e );</td>
<td>Vars(( e )) (( * ))</td>
</tr>
<tr>
<td>( x = M[e] );</td>
<td>Vars(( e )) (( * ))</td>
</tr>
<tr>
<td>( M[e_1] = e_2 );</td>
<td>Vars(( e_1 )) \cup Vars(( e_2 ))</td>
</tr>
</tbody>
</table>

(\( * \)) - given that \( x \) is truly live \( \Rightarrow \)
Example:

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

Example:

1. \( x = y + 1; \)
2. \( y, R \)
3. \( z = 2 \times x; \)
4. \( y, R \)
5. \( M[R] = y; \)
6. \( \emptyset \)
The Effects of Edges:

$$[\exists e] L = L$$
$$[\text{Pos}(e)] L = [\text{Neg}(e)] L = L \cup \text{Vars}(e)$$
$$[x = e] L = (L \setminus \{x\}) \cup \text{Vars}(e)$$
$$[x = M[e]] L = (L \setminus \{x\}) \cup \text{Vars}(e)$$
$$[M[e_1] = e_2] L = L \cup \text{Vars}(e_1) \cup \text{Vars}(e_2)$$

Example:

```
1 y, R
  x = y + 1;
2 y, R
  z = 2 * x;
3 y, R
  M[R] = y;
4

1 y, R
  ;
2 y, R
  ;
3 y, R
  M[R] = y;
4
```

The Effects of Edges:

$$[\exists e] L = L$$
$$[\text{Pos}(e)] L = [\text{Neg}(e)] L = L \cup \text{Vars}(e)$$
$$[x = e] L = (L \setminus \{x\}) \cup (x \in L) \text{? Vars}(e) : \emptyset$$
$$[x = M[e]] L = (L \setminus \{x\}) \cup (x \in L) \text{? Vars}(e) : \emptyset$$
$$[M[e_1] = e_2] L = L \cup \text{Vars}(e_1) \cup \text{Vars}(e_2)$$
**Note:**

- The effects of edges for truly live variables are more complicated than for live variables.
- Nonetheless, they are distributive!!

**The Effects of Edges:**

\[
\begin{align*}
&[\cdot]^2 L = L \\
&[\text{Pos}(e)]^2 L = [\text{Neg}(e)]^2 L = L \cup \text{Vars}(e) \\
&[x = e]^2 L = (L \setminus \{x\}) \cup (x \in L) \land \text{Vars}(e) : \emptyset \\
&[x = M[e]^2 ] L = (L \setminus \{x\}) \cup (x \in L) \land \text{Vars}(e) : \emptyset \\
&[M[e_1] = e_2]^2 L = L \cup \text{Vars}(e_1) \cup \text{Vars}(e_2)
\end{align*}
\]

**Note:**

- The effects of edges for truly live variables are more complicated than for live variables.
- Nonetheless, they are distributive!!

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

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

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

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

Note:

- True liveness detects more superfluous assignments than repeated liveness !!!!
1.3 Removing Superfluous Moves

Example:

This variable-variable assignment is obviously useless :-(

Example:

This variable-variable assignment is obviously useless :-(
Instead of \( y \), we could also store \( T \) :-(

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

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

Example:

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

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

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

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

---

Idea:

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

We use: $\mathcal{V} = \text{Expr} \rightarrow 2^{\text{Vars}}$ ...

```
[x = c]\mathcal{V} e' = \begin{cases} 
(V \cup \{x\}) & \text{if } e' = c \\
(V \setminus \{x\}) & \text{otherwise}
\end{cases}

[x = y]\mathcal{V} e' = \begin{cases} 
(V \cup \{x\}) & \text{if } y \in V e \\
(V \setminus \{x\}) & \text{otherwise}
\end{cases}

[x = e]\mathcal{V} e' = \begin{cases} 
\{x\} & \text{if } e' = e \\
(V \setminus \{x\}) & \text{otherwise}
\end{cases}

[x = M[e];]\mathcal{V} e' = (V \setminus \{x\})

[x = M[y];]\mathcal{V} e' = (V \setminus \{x\})

[x = M[e];]\mathcal{V} e' = \begin{cases} 
\emptyset & \text{if } e' = e \\
(V \setminus \{x\}) & \text{otherwise}
\end{cases}
```

$\mathcal{I}$ analogously for the diverse stores

// for Expr. Not an Id fun variable

\[\mathfrak{f} x\]
\[ [x = e]^{e'} = \begin{cases} (V(e) \cup \{x\}) & \text{if } e' = e \\ (V(e') \setminus \{x\}) & \text{otherwise} \end{cases} \]

\[ [x = y]^{e'} V e = \begin{cases} (V(e) \cup \{x\}) & \text{if } y \in V e \\ (V(e) \setminus \{x\}) & \text{otherwise} \end{cases} \]

\[ [x = e]^{e'} V e' = \begin{cases} \{x\} & \text{if } e' = e \\ (V(e') \setminus \{x\}) & \text{otherwise} \end{cases} \]

\[ [x = M[e]]^{e'} V e' = (V(e') \setminus \{x\}) \]

\[ [x = M[y]]^{e'} V e' = (V(e') \setminus \{x\}) \]

\[ [x = M[e]]^{e'} V e' = \begin{cases} \emptyset & \text{if } e' = e \\ (V(e') \setminus \{x\}) & \text{otherwise} \end{cases} \]

\[ \hat{V} e = \begin{cases} V e' & \text{if } e \in \mathbb{E}_{\lambda} \end{cases} \]

\[ \hat{V} e = \begin{cases} \emptyset & \text{otherwise} \end{cases} \]

\[ \emptyset \]

In the Example:

\[ T = x + 1; \]

\[ y = T; \]

\[ M[R] = y; \]

\[ \{x + 1 \mapsto (y, T)\} \]