Remark:

If $f : \mathbb{D}_1 \to \mathbb{D}_2$ is distributive, then also monotonic  :-)

Remark:

If \( f : D_1 \to D_2 \) is distributive, then also monotonic :-)

Obviously: \( a \sqsubseteq b \) iff \( a \sqcup b = b \).
Remark:

If \( f : \mathbb{D}_1 \rightarrow \mathbb{D}_2 \) is distributive, then also monotonic :-)

Obviously: \( a \sqsubseteq b \) iff \( a \sqcup b = b \).
From that follows:

\[
\begin{align*}
fb &= f(a \sqcup b) \\
    &= fa \sqcup fb \\
\implies fa &\sqsubseteq fb
\end{align*}
\]
Assumption: all $v$ are reachable from $\textit{start}$. 
Assumption: all $v$ are reachable from $\textit{start}$.

Then:

**Theorem** \hspace{1cm} Kildall 1972

If all effects of edges $[[k]]^\#$ are distributive, then: $I^*[v] = I[v]$ for all $v$. 

182

Has developed the operating system CP/M and GUIs for PCs.
Assumption: all $v$ are reachable from $\text{start}$.

Then:

**Theorem**

Kildall 1972

If all effects of edges $[[k]]^\#$ are distributive, then: $\mathcal{I}^*[v] = \mathcal{I}[v]$ for all $v$. 
Assumption: all $v$ are reachable from $start$. 

Then:

**Theorem** \[ \text{Kildall 1972} \]

If all effects of edges $[k]^\#$ are distributive, then: $I^*[v] = I[v]$ for all $v$.

**Proof:**

It suffices to prove that $I^*$ is a solution :-)

For this, we show that $I^*$ satisfies all constraints :-))
(1) We prove for $\text{start}$:

$$
\mathcal{I}^*\text{[start]} = \bigsqcup \{ \llbracket \pi \rrbracket \# d_0 \mid \pi : \text{start} \rightarrow^* \text{start} \}
\ni \llbracket \epsilon \rrbracket \# d_0
\ni d_0 \quad :-)$$
We prove for \( \text{start} \):

\[
\mathcal{I}^*[\text{start}] = \bigsqcup \{ [[\pi]]^\# d_0 \mid \pi : \text{start} \rightarrow^* \text{start} \}
\]
\[
\ni \quad [e]^\# d_0
\]
\[
\ni \quad d_0 \quad :-)
\]

For every \( k = (u, _, v) \) we prove:

\[
\mathcal{I}^*[v] = \bigsqcup \{ [[\pi]]^\# d_0 \mid \pi : \text{start} \rightarrow^* v \}
\]
\[
\ni \quad \bigsqcup \{ [[\pi']k]^\# d_0 \mid \pi' : \text{start} \rightarrow^* u \}
\]
\[
= \bigsqcup \{ [[k]]^\# (\bigsqcup \{ [[\pi']]^\# d_0 \mid \pi' : \text{start} \rightarrow^* u \}) \}
\]
\[
= [[k]]^\# (\mathcal{I}^*[u])
\]

since \( \{ \pi' \mid \pi' : \text{start} \rightarrow^* u \} \) is non-empty \( :-) \)
Warning:

- **Reachability** of all program points cannot be abandoned!

Consider:

\[
\mathbb{D} = \mathbb{N} \cup \{\infty\}
\]
Warning:

- **Reachability** of all program points cannot be abandoned!

Consider:

\[ D = \mathbb{N} \cup \{\infty\} \]

Then:

\[ I[2] = \text{inc} 0 = 1 \]
\[ I^*[2] = \bigcup \emptyset = 0 \]
Warning:

- **Reachability** of all program points cannot be abandoned!
  
  Consider:

  Example diagram:

  \[ D = \mathbb{N} \cup \{\infty\} \]

  Then:

  \[ I[2] = \text{inc}0 = 1 \]

  \[ I^*[2] = \bigcup \emptyset = 0 \]

- **Unreachable** program points can always be thrown away :-)

190
Summary and Application:

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

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

$\rightarrow$ The effects of edges of the analysis of availability of expressions are distributive:

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

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

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 :-)
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 $x$. 

196
Thereby, the set of all defined or used variables at an edge \( k = (_, lab, _{\text{lab}}) \) 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>Vars ( (e) )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>Neg ( (e) )</td>
<td>Vars ( (e) )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( x = e; )</td>
<td>Vars ( (e) )</td>
<td>( { x } )</td>
</tr>
<tr>
<td>( x = M[e]; )</td>
<td>Vars ( (e) )</td>
<td>( { x } )</td>
</tr>
<tr>
<td>( M[e_1] = e_2; )</td>
<td>Vars ( (e_1) \cup 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 dead at \( u \) along \( \pi \) (relative to \( X \)).

Example:

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

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

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

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

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

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

$[k]^\#$ 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 :-)
We verify that these definitions are meaningful :-)

\[ M[y] = x; \]

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

\[
M[y] = x; \quad \{x, y\} \quad \emptyset
\]
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 :-)

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

\[
\begin{array}{c}
1 \quad 2 \quad 3 \quad 4 \quad 5 \\
\emptyset \quad \{y\} \quad \{x, y\} \quad \emptyset
\end{array}
\]
We verify that these definitions are meaningful :-)
The set of variables which are live at $u$ then is given by:

$$L^*[u] = \bigcup\{[\pi]^\# X \mid \pi : u \rightarrow^* \text{stop}\}$$

... literally:

- The paths start in $u$ :-)
  \[\implies\text{ As partial ordering for } L \text{ we use } \subseteq = \subseteq.\]
- The set of variables which are live at program exit is given by the set $X$ :-)}
Transformation 2:

\[ x = e; \quad x \notin \mathcal{L}^*[v] \]

\[ x = M[e]; \quad x \notin \mathcal{L}^*[v] \]
Correctness Proof:

$\rightarrow$ Correctness of the effects of edges: If $L$ is the set of variables which are live at the exit of the path $\pi$, then $\llbracket \pi \rrbracket \downarrow L$ is the set of variables which are live at the beginning of $\pi$ :-)

$\rightarrow$ 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 :-)

$\rightarrow$ Correctness of the transformation: In any execution of the transformed programs, the live variables always receive the same values :-))

212
Computation of the sets $\mathcal{L}^*[u]$:

(1) Collecting constraints:

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

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

(1) Collecting constraints:

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

(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 $[k]^\dagger$ are distributive :-))

Warning: The information is propagated backwards !!!
Example:

\[ x = M[I]; \]
\[ y = 1; \]
\[ M[R] = y; \]
\[ Pos(x > 1) \]
\[ Neg(x > 1) \]

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

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

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

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

**Warning:**

Removal of assignments to dead variables may kill further variables:

```
x = y + 1;
z = 2 * x;
M[R] = y;
∅
```
The left-hand side of no assignment is **dead** :-)

**Warning:**

Removal of assignments to dead variables may kill further variables:

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

**Warning:**

Removal of assignments to dead variables may kill further variables:

1. $x = y + 1$
2. $x, y, R$
3. $M[R] = y$
4. $∅$
The left-hand side of no assignment is **dead** :-(

**Warning:**

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**  :-)

**Warning:**

Removal of assignments to dead variables may kill further variables:

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

```plaintext
x = y + 1;
M[R] = y;
```
The left-hand side of no assignment is dead :-) 

Warning:

Removal of assignments to dead variables may kill further variables:
The left-hand side of no assignment is **dead**  :-)  

**Warning:**  
Removal of assignments to dead variables may kill further variables:

\[
\begin{align*}
1 & \quad y, R \\
2 & \quad x = y + 1; \\
3 & \quad x, y, R \\
4 & \quad z = 2 \times x; \\
1 & \quad y, R \\
2 & \quad y, R \\
3 & \quad M[R] = y; \\
4 & \quad \emptyset \\
1 & \quad y, R \\
2 & \quad y, R \\
3 & \quad M[R] = y; \\
4 & \quad \emptyset \\
1 & \quad y, R \\
2 & \quad y, R \\
3 & \quad M[R] = y; \\
4 & \quad \emptyset
\end{align*}
\]
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 \( k = (\_ , \text{lab} , v) \) is defined as:

<table>
<thead>
<tr>
<th>( \text{lab} )</th>
<th>truely 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) ) ((*))</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 \( v \) \( :-) \)
Example:

1

\[ x = y + 1; \]

2

\[ z = 2 \ast x; \]

3

\[ M[R] = y; \]

4

\[ \emptyset \]
Example:

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

1

\[ x = y + 1; \]

2

\[ y, R \]

\[ z = 2 \times x; \]

3

\[ y, R \]

\[ M[R] = y; \]

4

\[ \emptyset \]
Example:

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

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