Theorem  

Assume $\mathcal{D}$ is a complete lattice. Then every monotonic function $f : \mathcal{D} \to \mathcal{D}$ has a least fixpoint $d_0 \in \mathcal{D}$.

Let $P = \{d \in \mathcal{D} \mid f d \sqsubseteq d\}$.

Then $d_0 = \bigcap P$.  

Theorem  Knaster – Tarski

Assume $\mathbb{D}$ is a complete lattice. Then every monotonic function $f : \mathbb{D} \to \mathbb{D}$ has a least fixpoint $d_0 \in \mathbb{D}$.

Let $P = \{d \in \mathbb{D} \mid f \cdot d \sqsubseteq d\}$.

Then $d_0 = \bigcap P$.

Proof:

(1) $d_0 \in P$:
Theorem  \text{Knaster – Tarski}

Assume $\mathbb{D}$ is a complete lattice. Then every \textit{monotonic} function $f : \mathbb{D} \to \mathbb{D}$ has a \underline{least fixpoint} $d_0 \in \mathbb{D}$.

Let $P = \{d \in \mathbb{D} \mid f d \sqsubseteq d\}$.

Then $d_0 = \bigcap P$.

\textbf{Proof:}

(1) $d_0 \in P$:

\begin{align*}
    f d_0 & \sqsubseteq f d \sqsubseteq d \quad \text{for all } d \in P \\
    \implies f d_0 & \quad \text{is a lower bound of } P \\
    \implies f d_0 & \sqsubseteq d_0 \quad \text{since } d_0 = \bigcap P \\
    \implies d_0 & \in P \quad \therefore
\end{align*}
(2) \[ f d_0 = d_0 : \]
\( f d_0 = d_0 : \)

\[ f d_0 \subseteq d_0 \quad \text{by (1)} \]

\[ \implies f(f d_0) \subseteq f d_0 \quad \text{by monotonicity of } f \]

\[ \implies f d_0 \in P \]

\[ \implies d_0 \subseteq f d_0 \quad \text{and the claim follows} \]
(2) \[ f d_0 = d_0 : \]

\[ f d_0 \subseteq d_0 \quad \text{by (1)} \]

\[ \implies f(f d_0) \subseteq f d_0 \quad \text{by monotonicity of } f \]

\[ \implies f d_0 \in P \]

\[ \implies d_0 \subseteq f d_0 \quad \text{and the claim follows} \]

(3) \[ d_0 \quad \text{is least fixpoint:} \]
(2) \[ f \ d_0 = d_0 : \]
\[ f \ d_0 \subseteq d_0 \quad \text{by (1)} \]
\[ \implies f(f \ d_0) \subseteq f \ d_0 \quad \text{by monotonicity of } f \]
\[ \implies f \ d_0 \in P \]
\[ \implies d_0 \subseteq f \ d_0 \quad \text{and the claim follows} \]

(3) \[ d_0 \quad \text{is least fixpoint:} \]
\[ f \ d_1 = d_1 \subseteq d_1 \quad \text{an other fixpoint} \]
\[ \implies d_1 \in P \]
\[ \implies d_0 \subseteq d_1 \quad \therefore \]
Remark:

The least fixpoint $d_0$ is in $P$ and a lower bound

$\implies d_0$ is the least value $x$ with $x \sqsupseteq f\ x$
Remark:

The least fixpoint $d_0$ is in $P$ and a lower bound $d_0$ is the least value $x$ with $x \sqsupseteq f x$

Application:

Assume $x_i \sqsupseteq f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \quad (*)$

is a system of constraints where all $f_i : \mathbb{D}^n \rightarrow \mathbb{D}$ are monotonic.
Remark:

The least fixpoint \( d_0 \) is in \( P \) and a lower bound
\[ \implies d_0 \text{ is the least value } x \text{ with } x \supseteq f \cdot x \]

Application:

Assume \( x_i \supseteq f_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n \) \((*)\)
is a system of constraints where all \( f_i : \mathbb{D}^n \rightarrow \mathbb{D} \) are monotonic.
\[ \implies \text{least solution of } (*) = \text{least fixpoint of } F \]
Example 1: \( \mathcal{D} = 2^U, \quad f(x) = x \cap a \cup b \)
Example 1: \( \mathbb{D} = 2^U, \quad f x = x \cap a \cup b \)

<table>
<thead>
<tr>
<th></th>
<th>( f^k \perp )</th>
<th>( f^k \top )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0 )</td>
<td>( \emptyset )</td>
<td>( U )</td>
</tr>
</tbody>
</table>
Example 1: \[ \mathcal{D} = 2^U, \quad f(x) = x \cap a \cup b \]

<table>
<thead>
<tr>
<th>(f)</th>
<th>(f^k \bot)</th>
<th>(f^k \top)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(\emptyset)</td>
<td>(U)</td>
</tr>
<tr>
<td>1</td>
<td>(b)</td>
<td>(a \cup b)</td>
</tr>
</tbody>
</table>
Example 1:

\[ \mathcal{D} = 2^U, \quad f(x) = x \cap a \cup b \]

<table>
<thead>
<tr>
<th></th>
<th>( f^k \bot )</th>
<th>( f^k \top )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>\emptyset</td>
<td>( U )</td>
</tr>
<tr>
<td>1</td>
<td>( b )</td>
<td>( a \cup b )</td>
</tr>
<tr>
<td>2</td>
<td>( b )</td>
<td>( a \cup b )</td>
</tr>
</tbody>
</table>
Conclusion:

Systems of inequations can be solved through fixpoint iteration, i.e., by repeated evaluation of right-hand sides.
Conclusion:

Systems of inequations can be solved through fixpoint iteration, i.e., by repeated evaluation of right-hand sides

Caveat: Naive fixpoint iteration is rather inefficient
Conclusion:

Systems of inequations can be solved through fixpoint iteration, i.e., by repeated evaluation of right-hand sides

Caveat: Naive fixpoint iteration is rather inefficient

Example:
Conclusion:

Systems of inequations can be solved through fixpoint iteration, i.e., by repeated evaluation of right-hand sides

Caveat: Naive fixpoint iteration is rather inefficient

Example:
Conclusion:

Systems of inequations can be solved through fixpoint iteration, i.e., by repeated evaluation of right-hand sides

Caveat: Naive fixpoint iteration is rather inefficient

Example:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>\emptyset</td>
<td>\emptyset</td>
</tr>
<tr>
<td>1</td>
<td>{1, x &gt; 1, x - 1}</td>
<td>{1}</td>
</tr>
<tr>
<td>2</td>
<td>Expr</td>
<td>{1, x &gt; 1, x - 1}</td>
</tr>
<tr>
<td>3</td>
<td>{1, x &gt; 1, x - 1}</td>
<td>{1, x &gt; 1, x - 1}</td>
</tr>
<tr>
<td>4</td>
<td>{1}</td>
<td>{1}</td>
</tr>
<tr>
<td>5</td>
<td>Expr</td>
<td>{1, x &gt; 1, x - 1}</td>
</tr>
</tbody>
</table>
Conclusion:

Systems of inequations can be solved through fixpoint iteration, i.e., by repeated evaluation of right-hand sides

Caveat: Naive fixpoint iteration is rather inefficient

Example:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>\emptyset</td>
<td>\emptyset</td>
<td>\emptyset</td>
</tr>
<tr>
<td>1</td>
<td>{1, y &gt; 1, x - 1}</td>
<td>{1}</td>
<td>{1}</td>
</tr>
<tr>
<td>2</td>
<td>(Expr)</td>
<td>{1, y &gt; 1, x - 1}</td>
<td>{1, y &gt; 1}</td>
</tr>
<tr>
<td>3</td>
<td>{1, y &gt; 1, x - 1}</td>
<td>{1, y &gt; 1, x - 1}</td>
<td>{1, y &gt; 1, x - 1}</td>
</tr>
<tr>
<td>4</td>
<td>{1}</td>
<td>{1}</td>
<td>{1}</td>
</tr>
<tr>
<td>5</td>
<td>(Expr)</td>
<td>{1, y &gt; 1, x - 1}</td>
<td>{1, y &gt; 1}</td>
</tr>
</tbody>
</table>
Conclusion:

Systems of inequations can be solved through fixpoint iteration, i.e., by repeated evaluation of right-hand sides

Caveat: Naive fixpoint iteration is rather inefficient

Example:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>\emptyset</td>
<td>\emptyset</td>
<td>\emptyset</td>
<td>\emptyset</td>
</tr>
<tr>
<td>1</td>
<td>{x &gt; 1, x - 1}</td>
<td>{1}</td>
<td>{1}</td>
<td>{1}</td>
</tr>
<tr>
<td>2</td>
<td>Expr</td>
<td>{1, x &gt; 1, x - 1}</td>
<td>{1, x &gt; 1}</td>
<td>{1, x &gt; 1}</td>
</tr>
<tr>
<td>3</td>
<td>{x &gt; 1, x - 1}</td>
<td>{1, x &gt; 1, x - 1}</td>
<td>{1, x &gt; 1, x - 1}</td>
<td>{1, x &gt; 1}</td>
</tr>
<tr>
<td>4</td>
<td>{1}</td>
<td>{1}</td>
<td>{1}</td>
<td>{1}</td>
</tr>
<tr>
<td>5</td>
<td>Expr</td>
<td>{1, x &gt; 1, x - 1}</td>
<td>{1, x &gt; 1}</td>
<td>{1, x &gt; 1}</td>
</tr>
</tbody>
</table>
Conclusion:
Systems of inequations can be solved through fixpoint iteration, i.e., by repeated evaluation of right-hand sides.

Caveat: Naive fixpoint iteration is rather inefficient.

Example:
Idea: Round Robin Iteration

Instead of accessing the values of the last iteration, always use the current values of unknowns.
Idea: Round Robin Iteration

Instead of accessing the values of the last iteration, always use the current values of unknowns

Example:
Idea: Round Robin Iteration

Instead of accessing the values of the last iteration, always use the current values of unknowns

Example:

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>∅</td>
</tr>
<tr>
<td>1</td>
<td>{1}</td>
</tr>
<tr>
<td>2</td>
<td>{1, x &gt; 1}</td>
</tr>
<tr>
<td>3</td>
<td>{1, x &gt; 1}</td>
</tr>
<tr>
<td>4</td>
<td>{1}</td>
</tr>
<tr>
<td>5</td>
<td>{1, x &gt; 1}</td>
</tr>
</tbody>
</table>
```
Idea: Round Robin Iteration

Instead of accessing the values of the last iteration, always use the current values of unknowns

Example:

\[
\begin{array}{|c|c|}
\hline
& 1 & 2 \\
\hline
0 & \emptyset & \\
1 & \{1\} & \\
2 & \{1, x > 1\} & \\
3 & \{1, x > 1\} & \\
4 & \{1\} & \\
5 & \{1, x > 1\} & \text{dito} \\
\hline
\end{array}
\]
The code for **Round Robin** Iteration in **Java** looks as follows:

```java
for (i = 1; i ≤ n; i++) xi = ⊥; 
do {
    finished = true;
    for (i = 1; i ≤ n; i++) {
        new = f_i(x_1, ..., x_n);
        if (!(x_i ⊆ new)) {
            finished = false;
            x_i = x_i ∪ new;
        }
    }
} while (!finished);
```
Caveat:

The efficiency of RR-iteration depends on the ordering of the unknowns

!!!
Caveat:

The efficiency of RR-iteration depends on the ordering of the unknowns !!!

Good:

\[ \rightarrow \text{ } u \text{ before } v, \text{ if } u \rightarrow^* v; \]

\[ \rightarrow \text{ entry condition before loop body} \]
Caveat:

The efficiency of RR-iteration depends on the ordering of the unknowns !!!

Good:

\( u \) before \( v \), if \( u \rightarrow^* v \);

entry condition before loop body

Bad:

e.g., post-order DFS of the CFG, starting at \( \text{start} \)
Inefficient Round Robin Iteration:

$5$

$y = 1$

$0$

$\text{Neg}(x > 1)$

$4$

$\text{Pos}(x > 1)$

$3$

$x \times y$

$2$

$x = x$

$1$

$0$

$1$

$2$

$3$

$4$

$5$
Inefficient Round Robin Iteration:

\[
\begin{align*}
&\text{Neg}(x > 1) \\
&\text{Pos}(x > 1) \\
&\quad 0 \\
&\quad 3 \\
&\quad 2 \\
&\quad 1 \\
&\quad 5 \\
&\quad y = 1; \\
&\quad y = x \times y; \\
&\quad x = x + 1;
\end{align*}
\]

\[
\begin{array}{|c|c|}
\hline
& 1 \\
\hline
0 & \text{Expr} \\
1 & \{1\} \\
2 & \{1, x - 1, x > 1\} \\
3 & \text{Expr} \\
4 & \{1\} \\
5 & \emptyset \\
\hline
\end{array}
\]
Inefficient Round Robin Iteration:

```
Neg(x > 1)  Pos(x > 1)

4

5
  y = 1;

0

3
  y = x * y;

2
  x = x 1;

1
```

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Expr</td>
<td>{1, x &gt; 1}</td>
</tr>
<tr>
<td>1</td>
<td>{1}</td>
<td>{1}</td>
</tr>
<tr>
<td>2</td>
<td>{1, x - 1, x &gt; 1}</td>
<td>{1, x - 1, x &gt; 1}</td>
</tr>
<tr>
<td>3</td>
<td>Expr</td>
<td>{1, x &gt; 1}</td>
</tr>
<tr>
<td>4</td>
<td>{1}</td>
<td>{1}</td>
</tr>
<tr>
<td>5</td>
<td>Ø</td>
<td>Ø</td>
</tr>
</tbody>
</table>
Inefficient Round Robin Iteration:

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

\[
\begin{array}{c}
4 \\
\downarrow \\
3 \\
\downarrow \\
2 \\
\downarrow \\
1 \\
\end{array}
\quad
\begin{array}{c}
y = 1; \\
y = x * y; \\
x = x - 1; \\
x = x \quad 1;
\end{array}
\]

\[
\begin{array}{|c|c|c|c|}
\hline
& 1 & 2 & 3 \\
\hline
0 & Expr & \{1\} & \{1, x > 1\} \\
1 & \{1\} & \{1\} & \{1\} \\
2 & \{1, x - 1, x > 1\} & \{1, x - 1, x > 1\} & \{1, x > 1\} \\
3 & Expr & \{1, x > 1\} & \{1, x > 1\} \\
4 & \{1\} & \{1\} & \{1\} \\
5 & \emptyset & \emptyset & \emptyset \\
\hline
\end{array}
\]
Inefficient Round Robin Iteration:

- \( y = 1; \)
- \( \text{Pos}(x > 1) \)
- \( x = x \times y; \)
- \( \text{Neg}(x > 1) \)
- \( x = x + 1; \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( Expr )</td>
<td>( {1, x &gt; 1} )</td>
<td>( {1, x &gt; 1} )</td>
<td>dito</td>
</tr>
<tr>
<td>1</td>
<td>( {1} )</td>
<td>( {1} )</td>
<td>( {1} )</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( {1, x - 1, x &gt; 1} )</td>
<td>( {1, x - 1, x &gt; 1} )</td>
<td>( {1, x &gt; 1} )</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( Expr )</td>
<td>( {1, x &gt; 1} )</td>
<td>( {1, x &gt; 1} )</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( {1} )</td>
<td>( {1} )</td>
<td>( {1} )</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td></td>
</tr>
</tbody>
</table>

\( \Rightarrow \) significantly less efficient
... end of background on: Complete Lattices

Final Question:

Why is a (or the least) solution of the constraint system useful?
... end of background on: Complete Lattices

Final Question:

Why is a (or the least) solution of the constraint system useful ???

For a complete lattice $\mathbb{D}$, consider systems:

$$\mathcal{I}[\text{start}] \supseteq d_0$$

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

where $d_0 \in \mathbb{D}$ and all $[k]^\# : \mathbb{D} \rightarrow \mathbb{D}$ are monotonic ...
... end of background on: Complete Lattices

Final Question:

Why is a (or the least) solution of the constraint system useful ???

For a complete lattice $\mathbb{D}$, consider systems:

\[
\begin{align*}
\mathcal{I}[\text{start}] & \cong d_0 \\
\mathcal{I}[v] & \cong [k]^\#(\mathcal{I}[u]) \quad k = (u, _, v) \quad \text{edge}
\end{align*}
\]

where $d_0 \in \mathbb{D}$ and all $[k]^\# : \mathbb{D} \to \mathbb{D}$ are monotonic ...

\[\Rightarrow\] Monotonic Analysis Framework
Wanted: **MOP**  (Merge Over all Paths)

\[ \mathcal{I}^*[v] = \bigsqcup \{ [\pi]^d_0 \mid \pi : start \rightarrow^* v \} \]
Wanted: MOP (Merge Over all Paths)

\[ \mathcal{I}^*[v] = \bigsqcup \{ \pi^* d_0 \mid \pi : \text{start} \rightarrow^* v \} \]

Theorem Kam, Ullman 1975

Assume \( \mathcal{I} \) is a solution of the constraint system. Then:

\[ \mathcal{I}[v] \supseteq \mathcal{I}^*[v] \quad \text{for every } v \]
Wanted: MOP (Merge Over all Paths)

\[ \mathcal{I}^*[v] = \bigsqcup \{ \mathcal{P}^\# d_0 \mid \pi: \text{start} \rightarrow^* v \} \]

Theorem Kam, Ullman 1975

Assume \( \mathcal{I} \) is a solution of the constraint system. Then:

\[ \mathcal{I}[v] \supseteq \mathcal{I}^*[v] \quad \text{for every} \quad v \]

In particular: \( \mathcal{I}[v] \supseteq [\pi]^\# d_0 \quad \text{for every} \quad \pi: \text{start} \rightarrow^* v \)
Disappointment:

Are solutions of the constraint system just upper bounds?
Disappointment:

Are solutions of the constraint system just upper bounds ???

Answer:

In general: yes
Disappointment:

Are solutions of the constraint system just upper bounds ???

Answer:

In general: yes

With the notable exception when all functions $[k]^{#}$ are distributive ...
The function $f : \mathbb{D}_1 \rightarrow \mathbb{D}_2$ is called

- **distributive**, if $f(\bigsqcup X) = \bigsqcup \{f x \mid x \in X\}$ for all $\emptyset \neq X \subseteq \mathbb{D}$;
- **strict**, if $f \perp = \perp$.
- **totally distributive**, if $f$ is distributive and strict.
The function \( f : \mathbb{D}_1 \rightarrow \mathbb{D}_2 \) is called

- **distributive**, if \( f (\bigcup X) = \bigcup \{ f x \mid x \in X \} \) for all \( \emptyset \neq X \subseteq \mathbb{D} \);
- **strict**, if \( f \bot = \bot \);
- **totally distributive**, if \( f \) is distributive and strict.

**Examples:**

- \( f x = x \cap a \cup b \) for \( a, b \subseteq U \).
The function $f : \mathbb{D}_1 \rightarrow \mathbb{D}_2$ is called

- **distributive**, if $f (\bigcup X) = \bigcup \{ f x \mid x \in X \}$ for all $\emptyset \neq X \subseteq \mathbb{D}$;
- **strict**, if $f \bot = \bot$.
- **totally distributive**, if $f$ is distributive and strict.

**Examples:**

- $f x = x \cap a \cup b$ for $a, b \subseteq U$.

  **Strictness:** $f \emptyset = a \cap \emptyset \cup b = b = \emptyset$ whenever $b = \emptyset$
The function \( f : \mathcal{D}_1 \rightarrow \mathcal{D}_2 \) is called

- **distributive**, if \( f \left( \bigcup X \right) = \bigcup \{ f x \mid x \in X \} \) for all \( \emptyset \neq X \subseteq \mathcal{D} \);
- **strict**, if \( f \perp = \perp \).
- **totally distributive**, if \( f \) is distributive and strict.

**Examples:**

- \( f x = x \cap a \cup b \) for \( a, b \subseteq U \).

**Strictness:** \( f \emptyset = a \cap \emptyset \cup b = b = \emptyset \) whenever \( b = \emptyset \)

**Distributivity:**

\[
\begin{align*}
f (x_1 \cup x_2) &= a \cap (x_1 \cup x_2) \cup b \\
&= a \cap x_1 \cup a \cap x_2 \cup b \\
&= f x_1 \cup f x_2
\end{align*}
\] :-}
\[ D_1 = D_2 = \mathbb{N} \cup \{ \infty \}, \quad \text{inc } x = x + 1 \]
\[ D_1 = D_2 = \mathbb{N} \cup \{\infty\}, \quad \text{inc } x = x + 1 \]

\textbf{Strictness:} \quad f \perp = \text{inc } 0 = 1 \quad \neq \quad \perp
\[ \mathcal{D}_1 = \mathcal{D}_2 = \mathbb{N} \cup \{\infty\}, \quad \text{inc} \ x = x + 1 \]

**Strictness:** \[ f \perp = \text{inc} \ 0 = 1 \neq \perp \]

**Distributivity:** \[ f \left( \bigsqcup X \right) = \bigsqcup \{x + 1 \mid x \in X\} \quad \text{for} \quad \emptyset \neq X \]
• \( \mathbb{D}_1 = \mathbb{D}_2 = \mathbb{N} \cup \{\infty\} \), \( \text{inc } x = x + 1 \)

**Strictness:** \( f \bot = \text{inc } 0 = 1 \neq \bot \)

**Distributivity:** \( f(\bigsqcup X) = \bigsqcup \{x + 1 | x \in X\} \) for \( \emptyset \neq X \)

• \( \mathbb{D}_1 = (\mathbb{N} \cup \{\infty\})^2 \), \( \mathbb{D}_2 = \mathbb{N} \cup \{\infty\} \), \( f(x_1, x_2) = x_1 + x_2 \)
\[ \mathbb{D}_1 = \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}, \quad \text{inc } x = x + 1 \]

**Strictness:** \[ f \perp = \text{inc } 0 = 1 \neq \perp \]

**Distributivity:** \[ f(\bigcup X) = \bigcup \{x + 1 \mid x \in X\} \quad \text{for} \quad \emptyset \neq X \]

\[ \mathbb{D}_1 = (\mathbb{N} \cup \{\infty\})^2, \quad \mathbb{D}_2 = \mathbb{N} \cup \{\infty\}, \quad f(x_1, x_2) = x_1 + x_2 : \]

**Strictness:** \[ f \perp = 0 + 0 = 0 \]
\[ \mathcal{D}_1 = \mathcal{D}_2 = \mathbb{N} \cup \{\infty\}, \quad \text{inc} \ x = x + 1 \]

**Strictness:** \[ f \perp = \text{inc} \ 0 = 1 \neq \perp \]

**Distributivity:** \[ f (\bigsqcup X) = \bigsqcup \{x + 1 \mid x \in X\} \quad \text{for} \quad \emptyset \neq X \]

\[ \mathcal{D}_1 = (\mathbb{N} \cup \{\infty\})^2, \quad \mathcal{D}_2 = \mathbb{N} \cup \{\infty\}, \quad f(x_1, x_2) = x_1 + x_2 : \]

**Strictness:** \[ f \perp = 0 + 0 = 0 \]

**Distributivity:**

\[ f ((1, 4) \bigsqcup (4, 1)) = f (4, 4) = 8 \]

\[ \neq 5 = f (1, 4) \bigsqcup f (4, 1) \quad :-) \]
Remark:

If $f : D_1 \rightarrow D_2$ is distributive, then also monotonic
Remark:

If $f : \mathcal{D}_1 \rightarrow \mathcal{D}_2$ is distributive, then also monotonic

Obviously: $a \subseteq b$ 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:

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

Then:

**Theorem**

If all effects of edges \([k]^\#\) are distributive, then:
\[
\mathcal{I}[v] = \mathcal{I}^*[v]
\]
for all \( v \).

\textit{Kildall 1972}
Assumption: all $v$ are reachable from $\textit{start}$.

Then:

**Theorem**

Kildall 1972

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

- Reachability of all program points is necessary! Consider:

\[
\begin{array}{c}
\text{7} \\
0 \quad 1 \quad 2
\end{array}
\]

where \( \mathbb{D} = \mathbb{N} \cup \{\infty\} \)
Caveat:

- **Reachability** of all program points is necessary! Consider:

\[
\begin{align*}
\mathcal{I}[2] &= \text{inc } 0 &= 1 \\
\mathcal{I}^*[2] &= \bigcup \emptyset &= 0
\end{align*}
\]

where \( \mathbb{D} = \mathbb{N} \cup \{\infty\} \).
Caveat:

- **Reachability** of all program points is necessary! Consider:

  \[
  \begin{align*}
  0 & \xrightarrow{\text{7}} 1 \\
  1 & \xrightarrow{\text{inc}} 2
  \end{align*}
  \]

  where \( \mathbb{D} = \mathbb{N} \cup \{\infty\} \)

  Then:

  \[
  \begin{align*}
  \mathcal{I}[2] &= \text{inc } 0 &= 1 \\
  \mathcal{I}^*[2] &= \bigcup \emptyset &= 0
  \end{align*}
  \]

- **Unreachable** program parts can be deleted.
Summary and Application:

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

→ The abstract edge effects in the analysis of availability of expressions are distributive:
Summary and Application:

→ The abstract edge effects in 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.