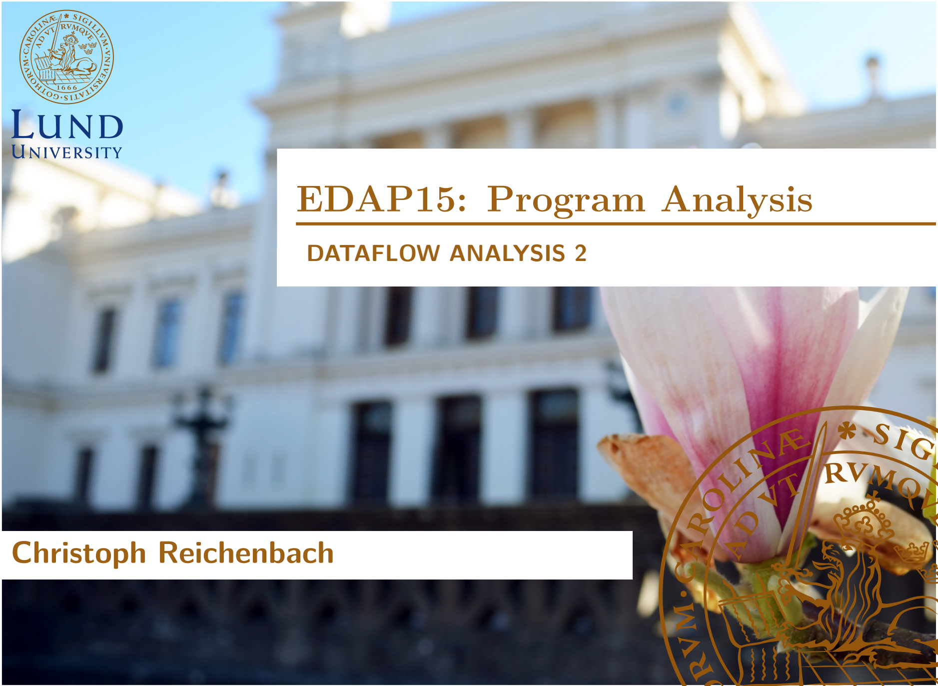


EDAP15: Program Analysis

DATAFLOW ANALYSIS 2

Christoph Reichenbach

Data Flow Analysis on CFGs

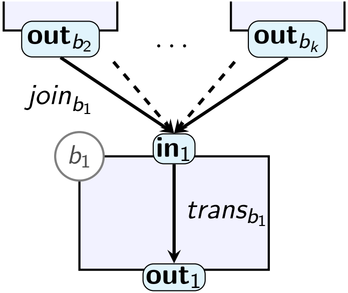
- $join_b$: *Join Function*
- $trans_b$: *Transfer Function*
- $\mathbf{in}_b$: knowledge at entrance of b

$$\mathbf{in}_{b_1} = join_{b_1}(\mathbf{out}_{b_2}, \ldots, \mathbf{out}_{b_k})$$

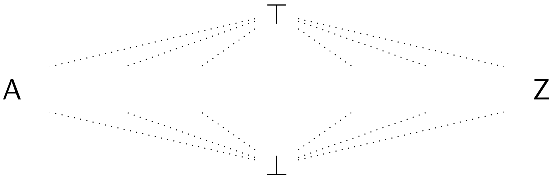
- $\mathbf{out}_b$: knowledge at exit of b

$$\mathbf{out}_{b_1} = trans_{b_1}(\mathbf{in}_{b_1})$$

- *Forward Analysis* shown here
- *Backward Analysis*: flip edge direction



Join and Transfer Functions



- L: Abstract Domain
  - Ordered by $(\sqsubseteq) \subseteq L \times L$

$$
\begin{array}{lll}
\top \in L & \text{for all } x: \quad x \sqsubseteq \top & \text{Top element} \\
\bot \in L & \text{for all } x: \quad \bot \sqsubseteq x & \text{Bottom element (optional)}
\end{array}
$$

- $trans_b : L \to L$
  - monotonic

$$
\begin{array}{ccc}
x & \sqsubseteq & y \\
 & \Downarrow & \\
trans_b(x) & \sqsubseteq & trans_b(y)
\end{array}
$$

- $join_b : L \times \ldots \times L \to L$
  - pointwise monotonic

$$
\begin{array}{ccc}
x & \sqsubseteq & y \\
 & \Downarrow & \\
join_b(z_1, \ldots, z_k, x, \ldots, z_n) & \sqsubseteq & join_b(z_1, \ldots, z_k, y, \ldots, z_n)
\end{array}
$$

Monotone Frameworks

Monotone Framework	Lattice
Abstract Domain	$L = \langle \mathcal{L}, \sqsubseteq, \sqcap, \sqcup \rangle$
$join_b(x_1, \ldots, x_n)$	$x_1 \sqcup \ldots \sqcup x_n$
	$x \sqcap y$ (Not needed)
'Unknown' start value	$\bot$
'Could be anything' end value	$\top$

- *Monotone Frameworks* (Killdall '77):
  - Lattice L of *finite height* (= satisfies Ascending Chain Condition)
  - Monotone $trans_b$
    - 'compatible' with semantics

$\Rightarrow$ Data flow analysis with Soundness and Termination

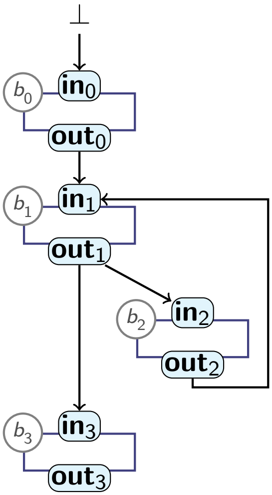
- Don't need $\sqcap$, so technically we only need a *Semilattice*.

Formalising our Naïve Algorithm

$$\begin{aligned}\mathbf{out}_0 &= \mathit{trans}_0(\bot)\\ \mathbf{out}_1 &= \mathit{trans}_1(\mathbf{out}_0 \sqcup \mathbf{out}_2)\\ \mathbf{out}_2 &= \mathit{trans}_2(\mathbf{out}_1)\\ \mathbf{out}_3 &= \mathit{trans}_3(\mathbf{out}_1)\end{aligned}$$

- Lattices $\mathbf{out}_0 : L_0, \ldots, \mathbf{out}_3 : L_3$
- Can build lattice for entire program:
  - $L_{0\ldots3} = L_0 \times L_1 \times L_2 \times L_3$
  - $\bot_{0\ldots3} = \langle \bot_0, \bot_1, \bot_2, \bot_3 \rangle$
  - Monotone transfer function:

$$\mathit{trans}_{0\ldots3}(\langle v_0, v_1, v_2, v_3 \rangle) = \left\langle \begin{array}{l} \mathit{trans}_0(v_0),\\ \mathit{trans}_1(v_0 \sqcup v_2),\\ \mathit{trans}_2(v_1),\\ \mathit{trans}_3(v_1) \end{array} \right\rangle$$



Reaching a Solution

- In general:
  - Program P:
    - "Program Lattice" L_P
    - $\bot_P$: initial analysis state
    - $trans_P$: Compute one step of naïve analysis
  - Repeat $trans_P$ until solution $fp_\bot$:

$$fp_\bot = trans_P^n(\bot_P)$$

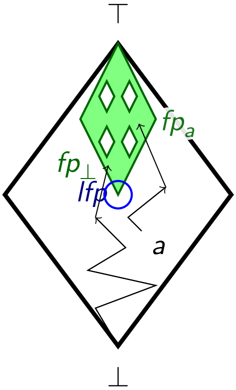
  - n is the minimum number of steps until we have a solution
- $fp_\bot$ is *Fixpoint* of $trans_P$:

$$fp_\bot = trans_P(fp_\bot)$$

- Fixpoint exists **iff** L_P satisfies Ascending Chain Condition

Cousot & Cousot (1979), based on Kleene (1952), based on Knaster & Tarski (1933)

Fixpoints



- Repeat $trans_P$ until we reach a fixpoint
- Can start from *any* point a
- Multiple fixpoints possible
  - Each is a *sound* solution (for *compatible* transfer functions)
  - Form a lattice (Knaster-Tarski, 1933)
- Least Fixpoint: Highest Precision

Value Range Analysis

'Find *value range* (interval of possible values) for x'

Teal

```
x := 1;
while ... {
  if ... {
    x := 4
  } else {
    x := 7
} }
```

- Multiple possible *sound* solutions:
  - $\top$
  - $[-99, 99]$
  - $[1, 10]$
  - $[1, 7]$
- All of these values are fixpoints
- $[1, 7]$ is *least fixpoint*

Summary

- **Monotone Frameworks**:
  - Combine:
    - Monotone transfer functions $trans_b$
    - Finite-Height Lattices

$$join_b(v_1, \ldots, v_k) = v_1 \sqcup \ldots \sqcup v_k$$

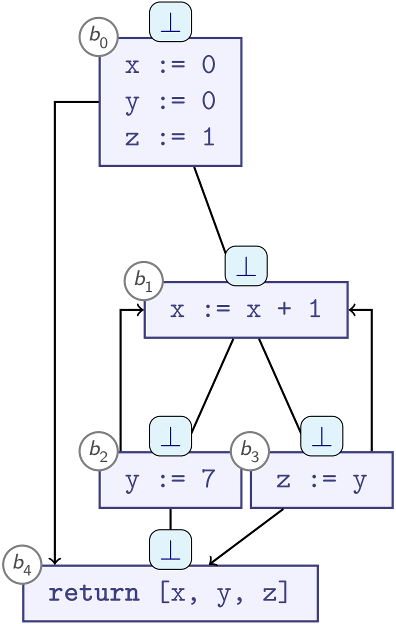
  - Guarantee:
    - Termination
    - Soundness
- With Monotone Frameworks, iterating $trans_b$ and $join_b$ produces **Fixpoint** (or *Fixed Point*)
  - Works from *any* starting point, possibly different fixpoint
    - Fixpoints form **Fixpoint Lattice**
    - **Least Fixpoint** (Bottom element) is *most precise solution*
- (Soundness only if $trans_b$ are *compatible*)

An Algorithm for Fixpoints

- So far: naïve algorithm for computing fixpoint
  - Produces a fixpoint
  - Keeps iterating *all* trans_b / join_b functions, even if nothing changed
- Optimise processing with *worklist*
  - Set-like datastructure:
    - **add** element (if not already present)
    - **contains** test: is element present?
    - **pop** element: remove and return one element
  - Tracks *what's left to be done*

$\Rightarrow$ "MFP" (Minimal Fixed Point) Algorithm
(*Does not always produce least fixpoint!*)

MFP Example:



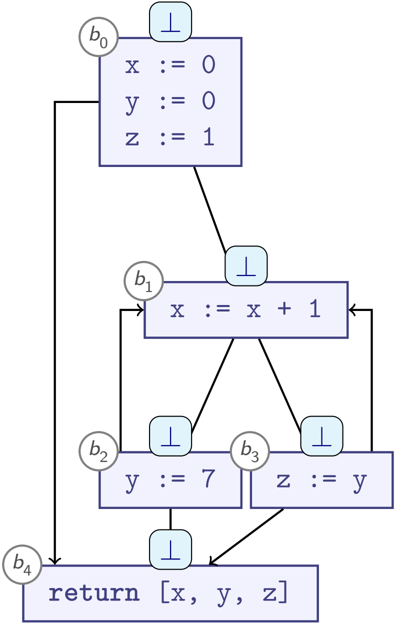
b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x+1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

Worklist

$b_0 \rightarrow b_1$
$b_0 \rightarrow b_4$
$b_1 \rightarrow b_2$
$b_1 \rightarrow b_3$
$b_2 \rightarrow b_4$
$b_2 \rightarrow b_1$
$b_3 \rightarrow b_4$
$b_3 \rightarrow b_1$

MFP Example:



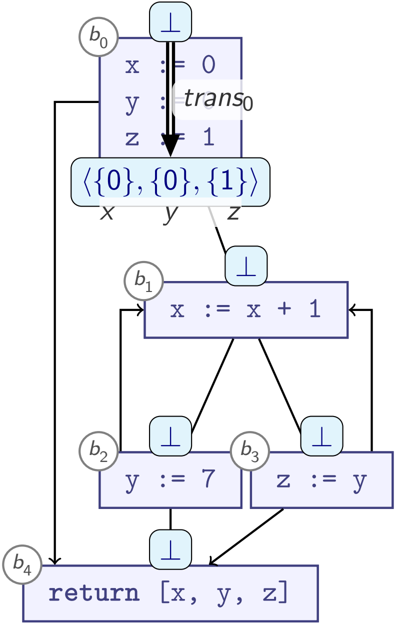
b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x + 1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

Worklist

$b_0 \to b_1$
$b_0 \to b_4$
$b_1 \to b_2$
$b_1 \to b_3$
$b_2 \to b_4$
$b_2 \to b_1$
$b_3 \to b_4$
$b_3 \to b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x+1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

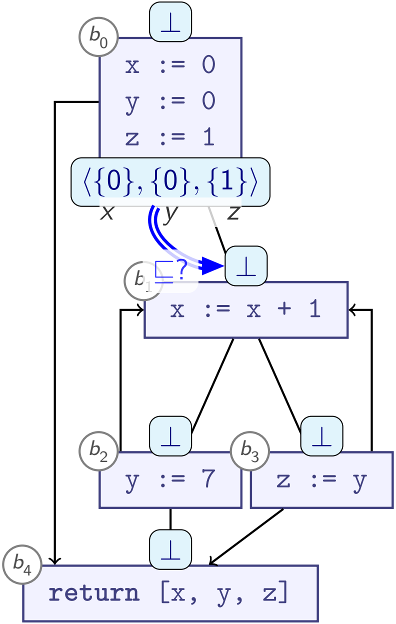
$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

For edge $b_o \to b_i$:

Worklist

- $b_0 \to b_1$
- $b_0 \to b_4$
- $b_1 \to b_2$
- $b_1 \to b_3$
- $b_2 \to b_4$
- $b_2 \to b_1$
- $b_3 \to b_4$
- $b_3 \to b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x + 1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

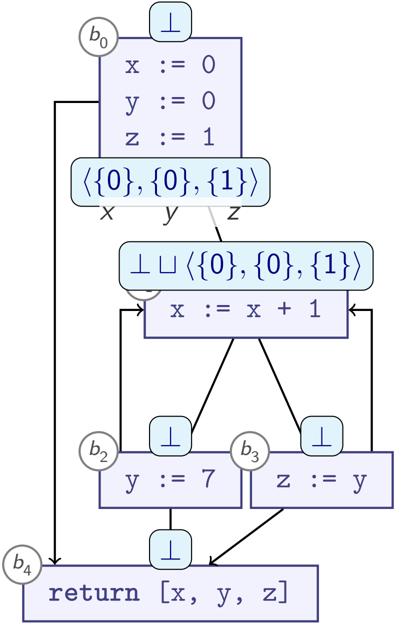
For edge $b_o \rightarrow b_i$:

- Is $\mathbf{out}_o \not\sqsubseteq \mathbf{in}_i$?

Worklist

$b_0 \rightarrow b_1$
$b_0 \rightarrow b_4$
$b_1 \rightarrow b_2$
$b_1 \rightarrow b_3$
$b_2 \rightarrow b_4$
$b_2 \rightarrow b_1$
$b_3 \rightarrow b_4$
$b_3 \rightarrow b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x + 1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

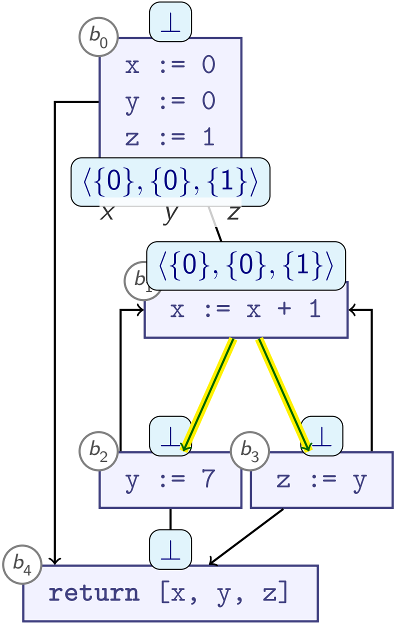
For edge $b_o \rightarrow b_i$:

- Is $\mathbf{out}_o \not\sqsubseteq \mathbf{in}_i$?
- **Yes**:
  - $\mathbf{in}_i := \mathbf{in}_i \sqcup \mathbf{out}_o$

Worklist

$b_0 \rightarrow b_1$
$b_0 \rightarrow b_4$
$b_1 \rightarrow b_2$
$b_1 \rightarrow b_3$
$b_2 \rightarrow b_4$
$b_2 \rightarrow b_1$
$b_3 \rightarrow b_4$
$b_3 \rightarrow b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x + 1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

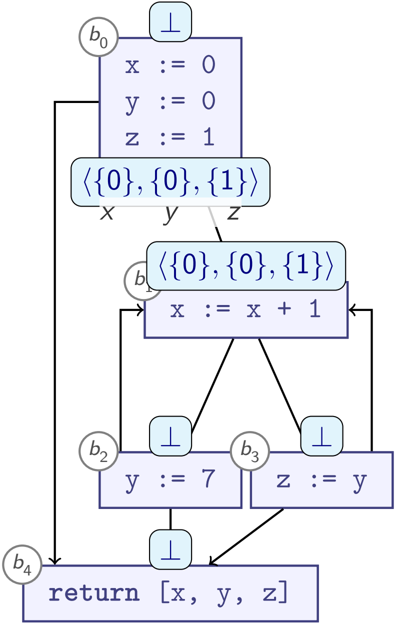
For edge $b_o \rightarrow b_i$:

- Is $\mathbf{out}_o \not\sqsubseteq \mathbf{in}_i$?
- **Yes**:
  - $\mathbf{in}_i := \mathbf{in}_i \sqcup \mathbf{out}_o$
  - Add all outgoing edges from b_o to worklist (if not already there)

Worklist

$b_0 \rightarrow b_1$
$b_0 \rightarrow b_4$
$b_1 \rightarrow b_2$
$b_1 \rightarrow b_3$
$b_2 \rightarrow b_4$
$b_2 \rightarrow b_1$
$b_3 \rightarrow b_4$
$b_3 \rightarrow b_1$

MFP Example:



b	inputs	$trans_b$		
		x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x+1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

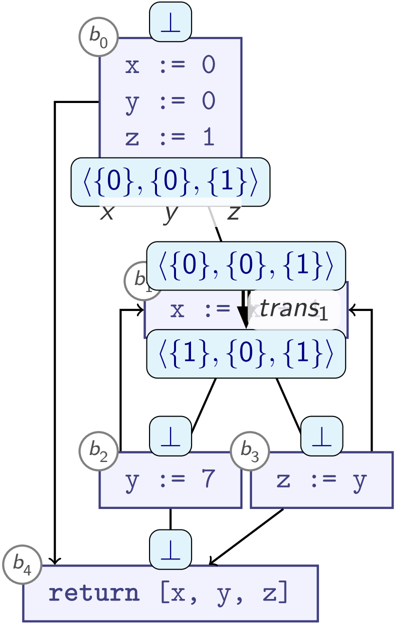
For edge $b_o \rightarrow b_i$:

- Is $\mathbf{out}_o \not\sqsubseteq \mathbf{in}_i$?
- **Yes**:
  - $\mathbf{in}_i := \mathbf{in}_i \sqcup \mathbf{out}_o$
  - Add all outgoing edges from b_o to worklist (if not already there)

Worklist

~~$b_0 \rightarrow b_1$~~
$b_0 \rightarrow b_4$
$b_1 \rightarrow b_2$
$b_1 \rightarrow b_3$
$b_2 \rightarrow b_4$
$b_2 \rightarrow b_1$
$b_3 \rightarrow b_4$
$b_3 \rightarrow b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x+1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

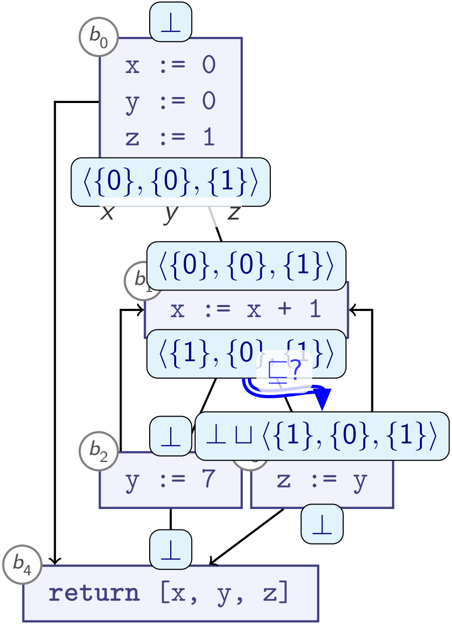
For edge $b_o \rightarrow b_i$:

- Is $\mathbf{out}_o \not\sqsubseteq \mathbf{in}_i$?
- **Yes**:
  - $\mathbf{in}_i := \mathbf{in}_i \sqcup \mathbf{out}_o$
  - Add all outgoing edges from b_o to worklist (if not already there)

Worklist

$b_0 \rightarrow b_4$
$b_1 \rightarrow b_2$
$b_1 \rightarrow b_3$
$b_2 \rightarrow b_4$
$b_2 \rightarrow b_1$
$b_3 \rightarrow b_4$
$b_3 \rightarrow b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x + 1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

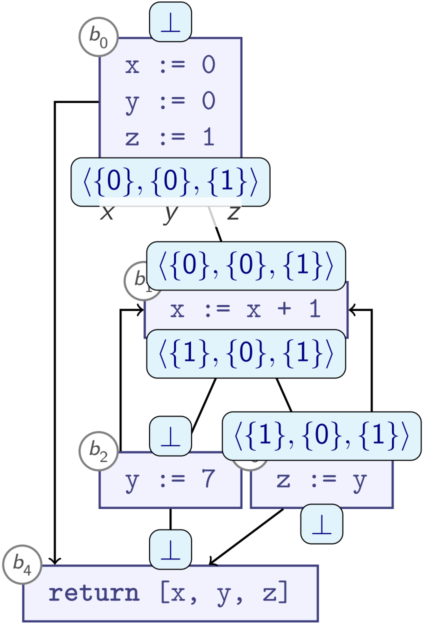
For edge $b_o \to b_i$:

- Is **out**$_o \not\sqsubseteq$ **in**$_i$?
- **Yes**:
  - **in**$_i$:= **in**$_i \sqcup$ **out**$_o$
  - Add all outgoing edges from b_o to worklist (if not already there)

Worklist

$b_0 \to b_4$
$b_1 \to b_2$
$b_1 \to b_3$
$b_2 \to b_4$
$b_2 \to b_1$
$b_3 \to b_4$
$b_3 \to b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x + 1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

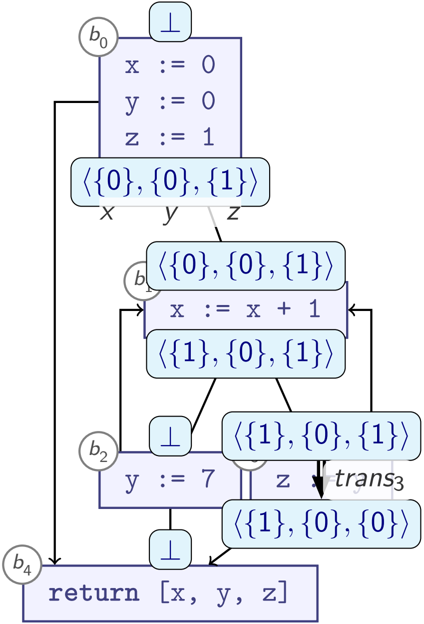
For edge $b_o \to b_i$:

- Is $\mathbf{out}_o \not\sqsubseteq \mathbf{in}_i$?
- **Yes**:
  - $\mathbf{in}_i := \mathbf{in}_i \sqcup \mathbf{out}_o$
  - Add all outgoing edges from b_o to worklist (if not already there)

Worklist

$b_0 \to b_4$
$b_1 \to b_2$
~~$b_1 \to b_3$~~
$b_2 \to b_4$
$b_2 \to b_1$
$b_3 \to b_4$
$b_3 \to b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x+1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

For edge $b_o \rightarrow b_i$:

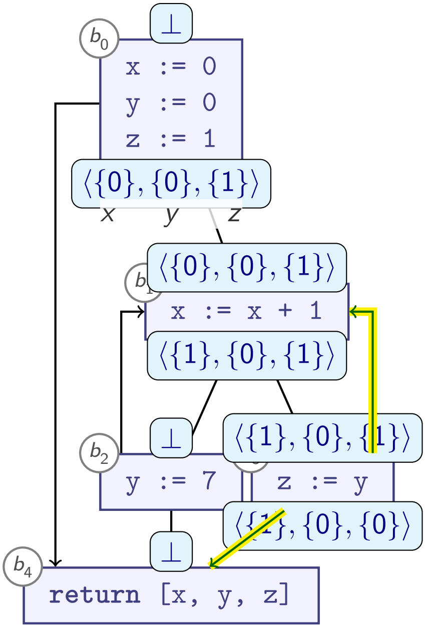
- Is **out**$_o \not\sqsubseteq$ **in**$_i$?
- **Yes**:
  - **in**$_i$:= **in**$_i \sqcup$ **out**$_o$
  - Add all outgoing edges from b_o to worklist (if not already there)

Worklist

$b_0 \rightarrow b_4$
$b_1 \rightarrow b_2$

$b_2 \rightarrow b_4$
$b_2 \rightarrow b_1$
$b_3 \rightarrow b_4$
$b_3 \rightarrow b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x + 1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

For edge $b_o \to b_i$:

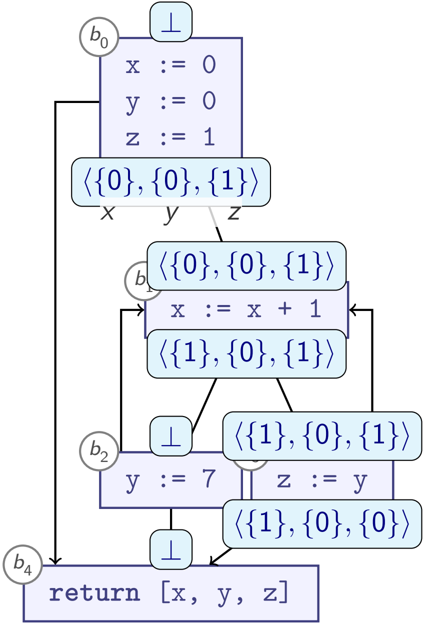
- Is $\mathbf{out}_o \not\sqsubseteq \mathbf{in}_i$?
- **Yes**:
  - $\mathbf{in}_i := \mathbf{in}_i \sqcup \mathbf{out}_o$
  - Add all outgoing edges from b_o to worklist (if not already there)

Worklist

$b_0 \to b_4$
$b_1 \to b_2$

$b_2 \to b_4$
$b_2 \to b_1$
$b_3 \to b_4$
$b_3 \to b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x+1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

For edge $b_o \to b_i$:

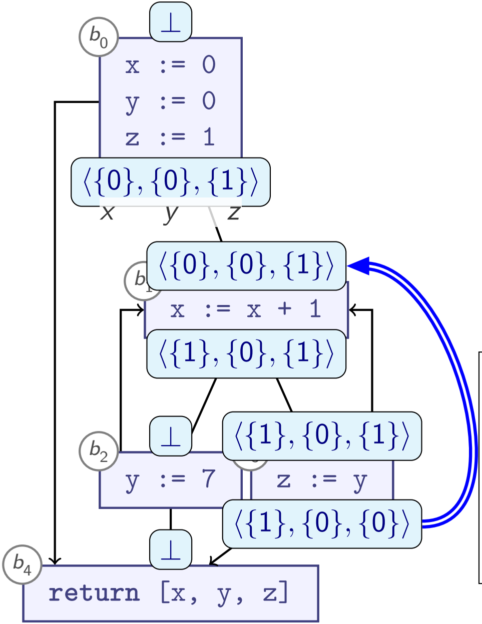
- Is **out**$_o \not\sqsubseteq$ **in**$_i$?
- **Yes**:
  - **in**$_i$:= **in**$_i \sqcup$ **out**$_o$
  - Add all outgoing edges from b_o to worklist (if not already there)

Worklist

$b_0 \to b_4$
$b_1 \to b_2$

$b_2 \to b_4$
$b_2 \to b_1$
$b_3 \to b_4$
$b_3 \to b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x+1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

For edge $b_o \to b_i$:

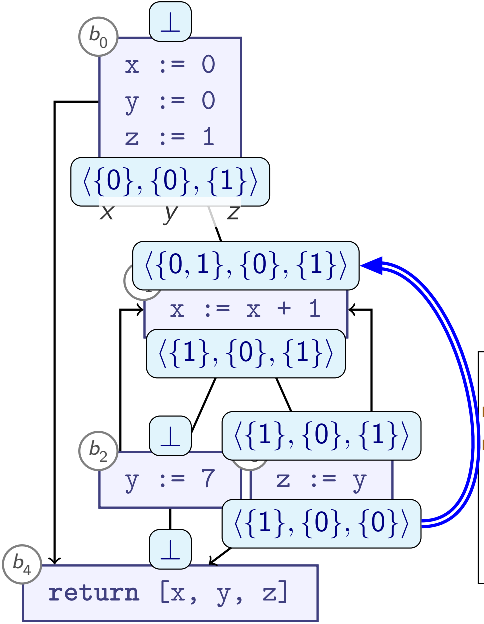
- Is $\mathbf{out}_o \not\sqsubseteq \mathbf{in}_i$?
- **Yes**:
  - $\mathbf{in}_i := \mathbf{in}_i \sqcup \mathbf{out}_o$
  - Add all outgoing edges from b_o to worklist (if not already there)

Worklist

$b_0 \to b_4$
$b_1 \to b_2$

$b_2 \to b_4$
$b_2 \to b_1$
$b_3 \to b_4$
$b_3 \to b_1$

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x+1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

For edge $b_o \to b_i$:

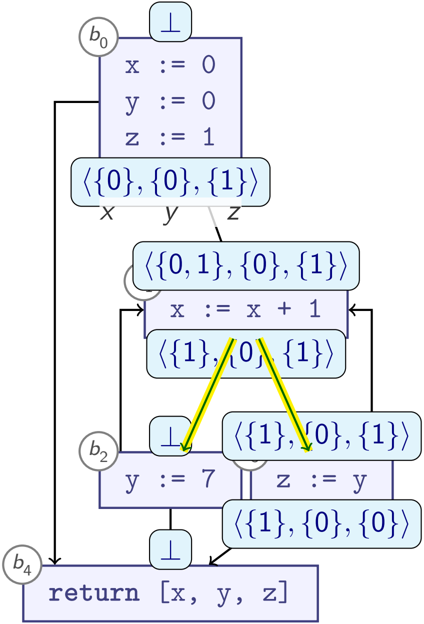
- Is $\mathbf{out}_o \not\sqsubseteq \mathbf{in}_i$?
- **Yes**:
  - $\mathbf{in}_i := \mathbf{in}_i \sqcup \mathbf{out}_o$
  - Add all outgoing edges from b_o to worklist (if not already there)

Worklist

$b_0 \to b_4$
$b_1 \to b_2$

$b_2 \to b_4$
$b_2 \to b_1$
$b_3 \to b_4$
~~$b_3 \to b_1$~~

MFP Example:



b	inputs	$trans_b$ x	y	z
b_0	$\emptyset$	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x+1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1}\rangle, \langle v_{x_2}, v_{y_2}, v_{z_2}\rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2}\rangle$$

For edge $b_o \to b_i$:

- Is $\mathbf{out}_o \not\sqsubseteq \mathbf{in}_i$?
- **Yes**:
  - $\mathbf{in}_i := \mathbf{in}_i \sqcup \mathbf{out}_o$
  - Add all outgoing edges from b_o to worklist (if no

Re-add previously removed edge

Worklist

$b_0 \to b_4$
$b_1 \to b_2$

$b_2 \to b_4$
$b_2 \to b_1$
$b_3 \to b_4$

$b_1 \to b_3$

The MFP Algorithm

```
Procedure MFP(⊥, ⊔, ⊑, CFG, trans_, is-backward):
begin
  if is-backward then reverse edges(CFG);
  worklist := edges(CFG); -- edges that we need to look at
  foreach n ∈ nodes(CFG) do
    in[n] := ⊥;            -- state of the analysis
  done
  while not empty(worklist) do
    ⟨n, n′⟩ := pop(worklist);    -- Edge n → n′
        -- OPTIONAL: cache out[n] = trans_n(in[n]) here
    if trans_n(in[n]) ⋢ in[n′] then begin
      in[n′] := in[n′] ⊔ trans_n(in[n]);
      foreach n″ ∈ successor-nodes(CFG, n′) do
        push(worklist, ⟨n′, n″⟩);
      done
    end
  done
  return in;
end
```

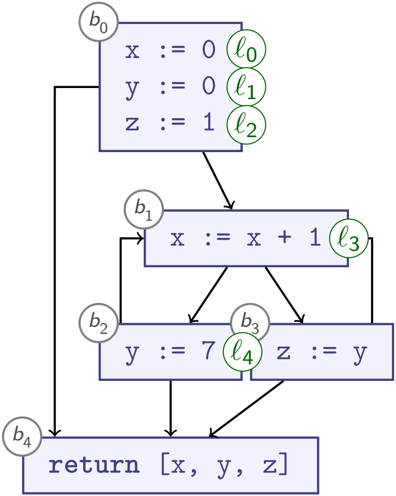
Worklist allows focussing effort!

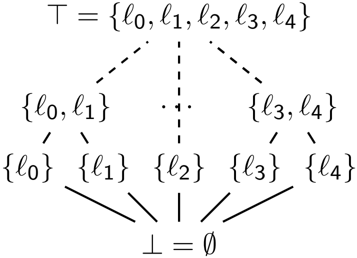
Summary: MFP Algorithm

- **Product Lattice** allows analysing multiple variables at once
- Compute data flow analysis:
  - Initialise all nodes with $\bot$
  - Repeat until nothing changes any more:
    - Apply transfer function
    - Propagate changes along control flow graph
    - Apply $\sqcup$
- Compute **fixpoint**
- Use **worklist** to increase efficiency
- Distinction: Forward/Backward analyses

MFP revisited

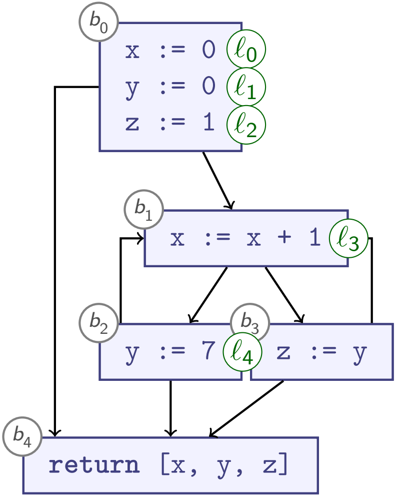
Consider **Reaching Definitions** again, with different lattice:





- All subsets of $\{\ell_0, \ldots, \ell_4\}$
- Finite height
- $\sqcup = \cup$

MFP revisited: Transfer Functions



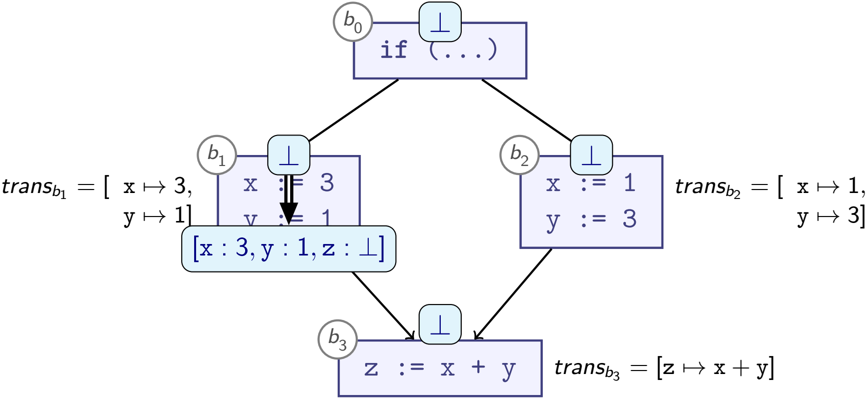
$$
\begin{aligned}
trans_{b_0} &= [\; x \mapsto \{\ell_0\},\\
&\quad\;\; y \mapsto \{\ell_1\},\\
&\quad\;\; z \mapsto \{\ell_2\}]\\
trans_{b_1} &= [x \mapsto \{\ell_3\}]\\
trans_{b_2} &= [y \mapsto \{\ell_4\}]\\
trans_{b_3} &= [z \mapsto y]
\end{aligned}
$$

MFP solution

$$
\begin{aligned}
x &\mapsto \{\ell_0, \ell_3\}\\
y &\mapsto \{\ell_1, \ell_4\}\\
z &\mapsto \{\ell_1, \ell_2, \ell_4\}
\end{aligned}
$$

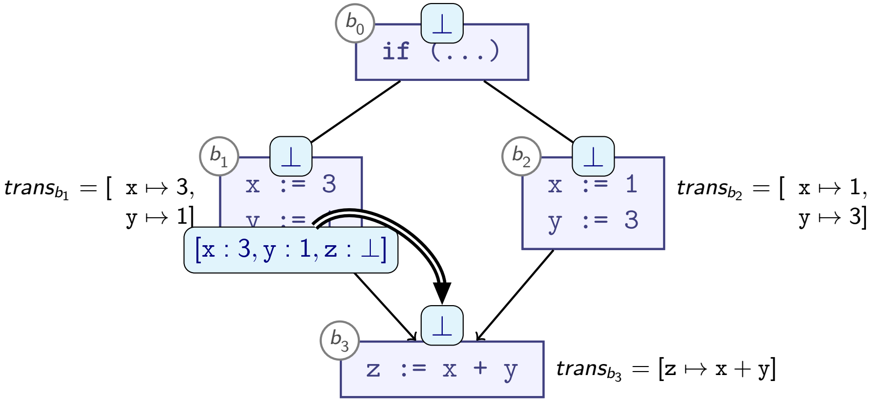
- Least Fixpoint!
- Do we always get LFP from MFP?

Another Example



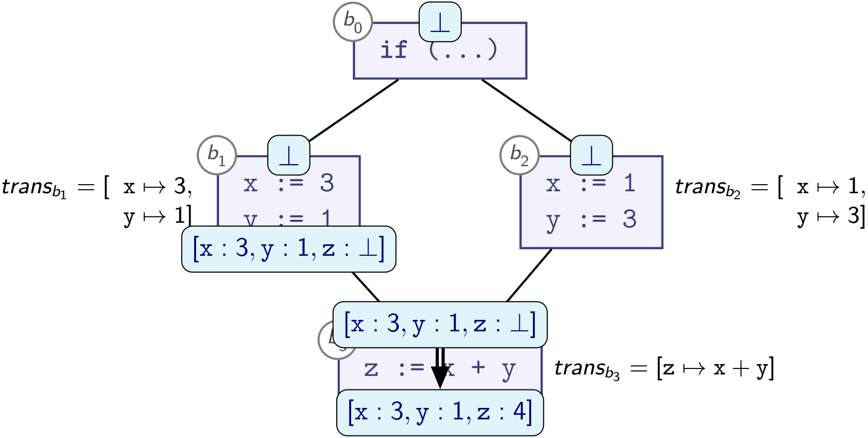
- Lattice: $\mathbb{Z}_\bot^\top$

Another Example



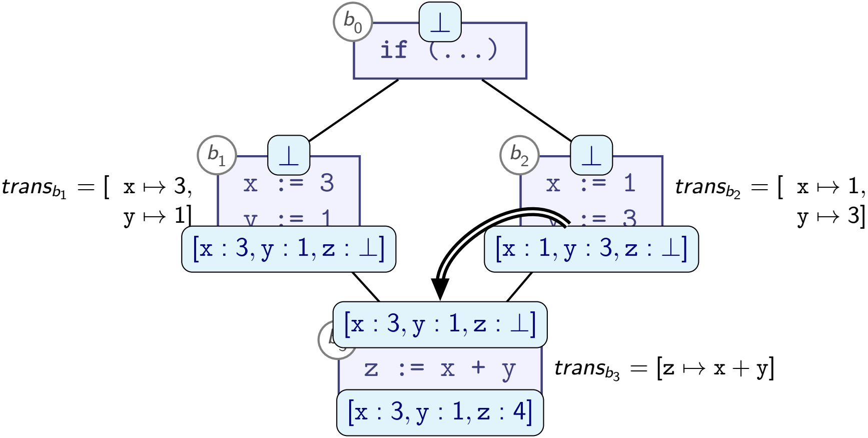
- Lattice: $\mathbb{Z}_{\bot}^{\top}$

Another Example



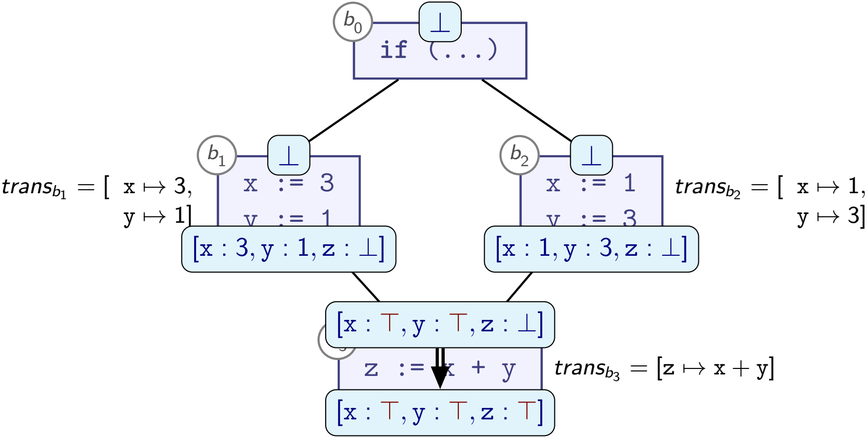
- Lattice: $\mathbb{Z}_\bot^\top$

Another Example



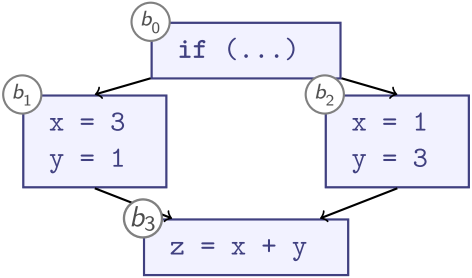
- Lattice: $\mathbb{Z}_{\bot}^{\top}$
  - $1 \sqcup 3 = \top = 3 \sqcup 1$

Another Example



- Lattice: $\mathbb{Z}_\bot^\top$
  - $1 \sqcup 3 = \top = 3 \sqcup 1$
- MFP **does** compute the Least Fixpoint in our equations...
- ...but the fixpoint is worse than expected!

Execution paths



- Idea: Let's consider all *paths* through the program:

$$
\begin{aligned}
path_{b_0} &= \{[]\} \\
path_{b_1} &= \{[b_0]\} \\
path_{b_2} &= \{[b_0]\} \\
path_{b_3} &= \{[b_0, b_1], [b_0, b_2]\}
\end{aligned}
$$

The MOP algorithm for Dataflow Analysis

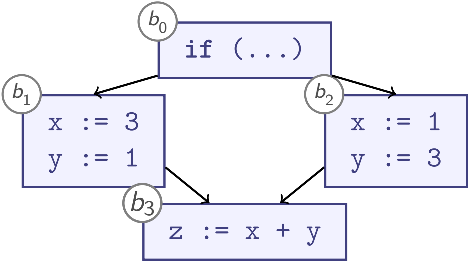
- Compute the MOP ('meet-over-all-paths') solution:
  - Iterate over all paths $[p_0, \ldots, p_k]$ in $path_{b_i}$
  - Compute *precise* result for that path
  - Merge (i.e., join, $\sqcup$) with all other precise results

$$\mathbf{out}_{b_i} = \bigsqcup_{[p_0,\ldots,p_k] \in path_{b_i}} trans_{b_i} \circ trans_{p_k} \circ \cdots \circ trans_{p_0}(\bot)$$

Notation: (*function composition*)

$$(f \circ g)(x) = f(g(x))$$

MOP vs MFP: Example



Transfer functions

$$
\begin{aligned}
trans_{b_0} &= id \\
trans_{b_1} &= [x \mapsto 3][y \mapsto 1] \\
trans_{b_2} &= [x \mapsto 1][y \mapsto 3] \\
trans_{b_3} &= [z \mapsto x + y]
\end{aligned}
$$

Paths

$$
\begin{aligned}
path_{b_0} &= \{[]\} \\
path_{b_1} &= \{[b_0]\} \\
path_{b_2} &= \{[b_0]\} \\
path_{b_3} &= \{[b_0, b_1], [b_0, b_2]\}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{out}_{b_3} &= ([z \mapsto x + y][x \mapsto 3][y \mapsto 1](\bot)) \sqcup ([z \mapsto x + y][x \mapsto 1][y \mapsto 3](\bot)) \\
&= \{z \mapsto 3 + 1, x \mapsto 3, y \mapsto 1\} \sqcup \{z \mapsto 1 + 3, x \mapsto 1, y \mapsto 3\} \\
&= \{z \mapsto 4, x \mapsto \top, y \mapsto \top\}
\end{aligned}
$$

MOP vs MFP

	MOP	**MFP**
Soundness	sound	sound
Precision	*maximal*	*sometimes lower*
Decidability	*undecidable*	*decidable*

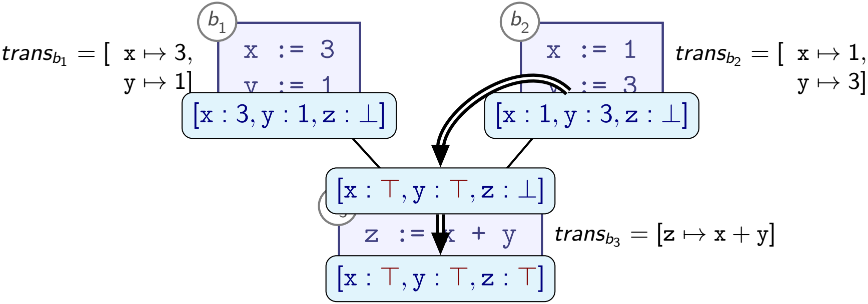
- MOP: Merge Over all Paths
  (Originally: "Meet Over all Paths", but we use the Join operator)
- MFP: Maximal Fixed Point

Summary

- $path_b$: Set of all paths from program start to b
- MOP: alternative to MFP (theoretically)
  - Termination not guaranteed
  - May be more precise
  - Idea:
    - Enumerate all paths to basic block
    - Compute transfer functions over paths individually
    - Join

Why is MFP *sometimes* as good as MOP?

MFP vs the Least Fixpoint



- MFP is *sometimes* equal to MOP
- Challenge:

$$trans_b(x \sqcup y) \sqsupset trans_b(x) \sqcup trans_b(y)$$

- join-before-transfer: overapproximate before we can reconcile!

Distributive Frameworks

A *Monotone Framework* is:

- Lattice $L = \langle \mathcal{L}, \sqsubseteq, \sqcap, \sqcup \rangle$
- L has finite height (Ascending Chain Condition)
- All $trans_b$ are monotonic
- Guarantees a Fixpoint

A *Distributive Framework* is:

- A Monotone Framework, where additionally:
- $trans_b$ distributes over $\sqcup$:

$$trans_b(x \sqcup y) = trans_b(x) \sqcup trans_b(y)$$

  for all programs and all x, y, b
- Guarantees that MFP gives same Fixpoint as MOP

Distributive Problems

- Monotonic:

$$trans_b(x \sqcup y) \sqsupseteq trans_b(x) \sqcup trans_b(y)$$

- Distributive:

$$trans_b(x \sqcup y) = trans_b(x) \sqcup trans_b(y)$$

- Many analyses fit distributive framework
- Known *counter-example*: transfer functions on $\mathbb{Z}_{\perp}^{\top}$:
  - $[z \mapsto x + y]$
  - Generally:
    - depends on ≥ 2 independent inputs
    - can produce same output for different inputs

Summary

- **Distributive Frameworks** are *Monotone Frameworks* with additional property:

$$trans_b(x \sqcup y) = trans_b(x) \sqcup trans_b(y)$$

  for all programs and all x, y, b
- In Distributive Frameworks, MFP produces same least Fixpoint as for MOP
- Some analyses (Gen/Kill analyses, discussed later) are always distributive