



LUND
UNIVERSITY

EDAP15: Program Analysis

DATAFLOW ANALYSIS 2

Christoph Reichenbach



Data Flow Analysis on CFGs

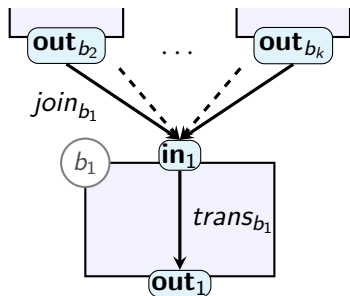
- ▶ $join_b$: Join Function
- ▶ $trans_b$: Transfer Function
- ▶ in_b :

$$in_{b_1} = join_{b_1}(out_{b_2}, \dots, out_{b_k})$$

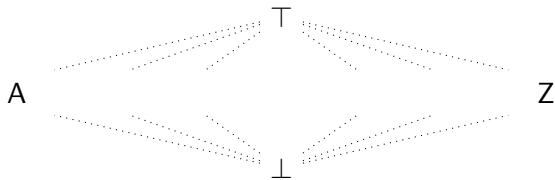
- ▶ out_b :

$$out_{b_1} = trans_{b_1}(in_{b_1})$$

- ▶ Forward Analysis
- ▶ Backward Analysis

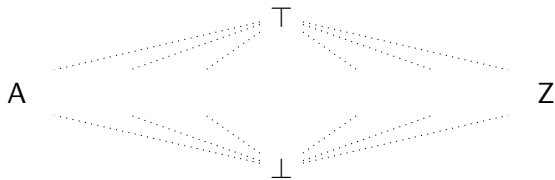


Join and Transfer Functions



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

Join and Transfer Functions



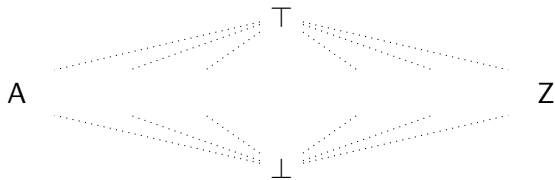
- ▶ L : Abstract Domain

- ▶ Ordered by $(\sqsubseteq) \subseteq L \times L$

- $\top \in L$ for all x : $x \sqsubseteq \top$ Top element

- $\perp \in L$ for all x : $\perp \sqsubseteq x$ Bottom element (optional)

Join and Transfer Functions



► L : Abstract Domain

- Ordered by $(\sqsubseteq) \subseteq L \times L$

$\top \in L$ for all x : $x \sqsubseteq \top$ Top element

$\perp \in L$ for all x : $\perp \sqsubseteq x$ Bottom element (optional)

► $trans_b : L \rightarrow L$

- monotonic

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

► $join_b : L \times \dots \times L \rightarrow L$

- pointwise monotonic

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

Monotone Frameworks

Monotone Framework

Abstract Domain

$join_b(x_1, \dots, x_n)$

'Unknown' start value

'Could be anything' end value

Lattice

$L = \langle \mathcal{L}, \sqsubseteq, \sqcap, \sqcup \rangle$

$x_1 \sqcup \dots \sqcup x_n$

$x \sqcap y$

\perp

\top

Monotone Frameworks

Monotone Framework

Abstract Domain

$join_b(x_1, \dots, x_n)$

'Unknown' start value

'Could be anything' end value

Lattice

$L = \langle \mathcal{L}, \sqsubseteq, \sqcap, \sqcup \rangle$

$x_1 \sqcup \dots \sqcup x_n$

$x \sqcap y$ (Not needed)

\perp

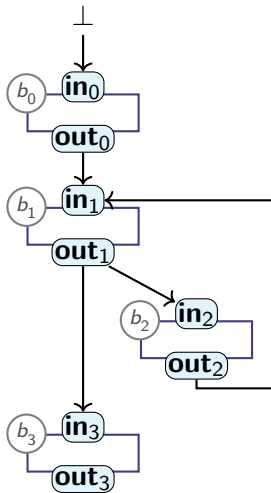
\top

Monotone Frameworks

Monotone Framework	Lattice
Abstract Domain	$L = \langle \mathcal{L}, \sqsubseteq, \sqcap, \sqcup \rangle$
$join_b(x_1, \dots, x_n)$	$x_1 \sqcup \dots \sqcup x_n$
'Unknown' start value	\perp
'Could be anything' end value	\top
▶ <i>Monotone Frameworks</i> (Killdall '77):	
▶ Lattice L of <i>finite height</i> (= satisfies Ascending Chain Condition)	
▶ Monotone $trans_b$	
▶ 'compatible' with semantics	
⇒ Data flow analysis with Soundness and Termination	

Formalising our Naïve Algorithm

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



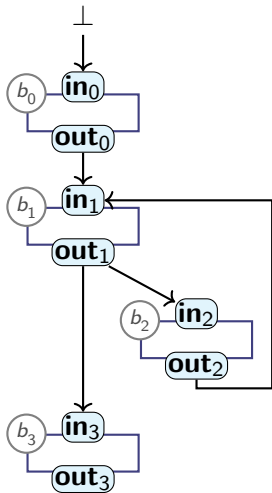
Formalising our Naïve Algorithm

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

► Lattices $\mathbf{out}_0 : L_0, \dots, \mathbf{out}_3 : L_3$

► Can build:

- $L_{0\dots3} = L_0 \times L_1 \times L_2 \times L_3$
- $\perp_{0\dots3} = \langle \perp_0, \perp_1, \perp_2, \perp_3 \rangle$



Formalising our Naïve Algorithm

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

► Lattices $\mathbf{out}_0 : L_0, \dots, \mathbf{out}_3 : L_3$

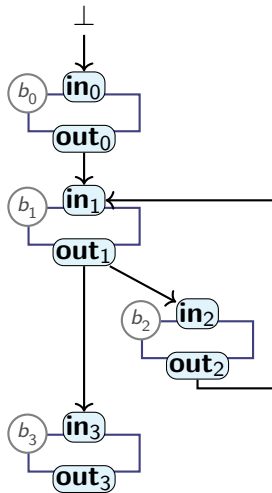
► Can build:

► $L_{0\dots3} = L_0 \times L_1 \times L_2 \times L_3$

► $\perp_{0\dots3} = \langle \perp_0, \perp_1, \perp_2, \perp_3 \rangle$

► transfer function:

$$\mathit{trans}_{0\dots3}(\langle v_0, v_1, v_2, v_3 \rangle) = \left\langle \begin{array}{l} \mathit{trans}_0(v_0), \\ \mathit{trans}_1(v_0 \sqcup v_2), \\ \mathit{trans}_2(v_1), \\ \mathit{trans}_3(v_1) \end{array} \right\rangle$$



Formalising our Naïve Algorithm

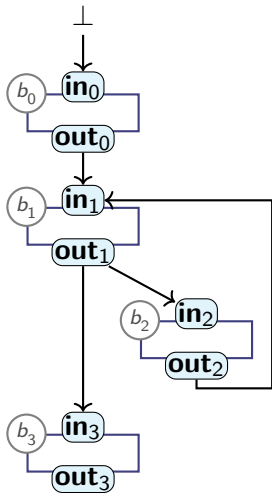
$$\begin{aligned}\mathbf{out}_0 &= \mathit{trans}_0(\perp) \\ \mathbf{out}_1 &= \mathit{trans}_1(\mathbf{out}_0 \sqcup \mathbf{out}_2) \\ \mathbf{out}_2 &= \mathit{trans}_2(\mathbf{out}_1) \\ \mathbf{out}_3 &= \mathit{trans}_3(\mathbf{out}_1)\end{aligned}$$

► Lattices $\mathbf{out}_0 : L_0, \dots, \mathbf{out}_3 : L_3$

► Can build:

- $L_{0\dots3} = L_0 \times L_1 \times L_2 \times L_3$
- $\perp_{0\dots3} = \langle \perp_0, \perp_1, \perp_2, \perp_3 \rangle$
- Monotone transfer function:

$$\mathit{trans}_{0\dots3}(\langle v_0, v_1, v_2, v_3 \rangle) = \left\langle \begin{array}{l} \mathit{trans}_0(v_0), \\ \mathit{trans}_1(v_0 \sqcup v_2), \\ \mathit{trans}_2(v_1), \\ \mathit{trans}_3(v_1) \end{array} \right\rangle$$



Reaching a Solution

- ▶ In general:
 - ▶ Program P :
 - ▶ “Program Lattice” L_P
 - ▶ \perp_P
 - ▶ $trans_P$: Compute one step of naïve analysis
 - ▶ Repeat $trans_P$:

$$fp_{\perp} = trans_P^n(\perp_P)$$

Reaching a Solution

- ▶ In general:
 - ▶ Program P :
 - ▶ “Program Lattice” L_P
 - ▶ \perp_P
 - ▶ $trans_P$: Compute one step of naïve analysis
 - ▶ Repeat $trans_P$:

$$fp_{\perp} = trans_P^n(\perp_P)$$

- ▶ fp_{\perp} is *Fixpoint* of $trans_P$:

$$fp_{\perp} = trans_P(fp_{\perp})$$

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

Reaching a Solution

- ▶ In general:
 - ▶ Program P :
 - ▶ “Program Lattice” L_P
 - ▶ \perp_P
 - ▶ $trans_P$: Compute one step of naïve analysis
 - ▶ Repeat $trans_P$:

$$fp_{\perp} = trans_P^n(\perp_P)$$

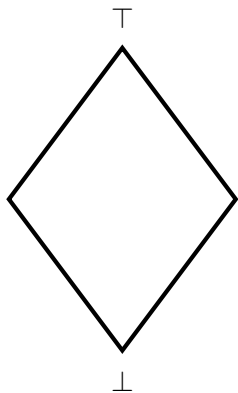
- ▶ fp_{\perp} is *Fixpoint* of $trans_P$:

$$fp_{\perp} = trans_P(fp_{\perp})$$

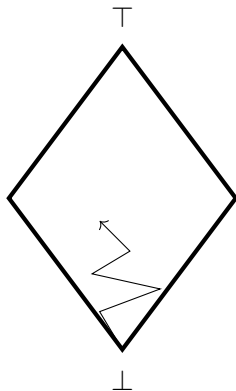
- ▶ Fixpoint exists **iff** L_P satisfies Ascending Chain Condition

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

Fixpoints

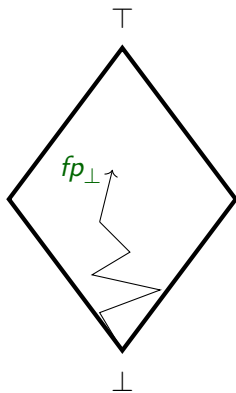


Fixpoints



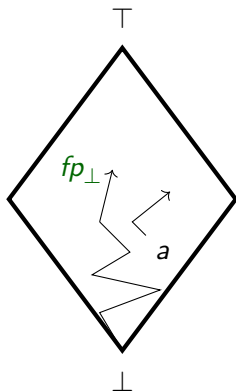
- ▶ Repeat *trans_P*

Fixpoints



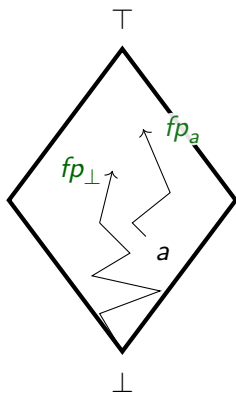
- ▶ Repeat $trans_P$ until fixpoint

Fixpoints



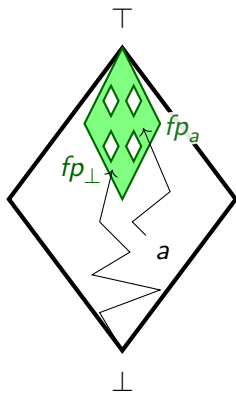
- ▶ Repeat $trans_P$ until fixpoint
- ▶ Can start from *any* point a

Fixpoints



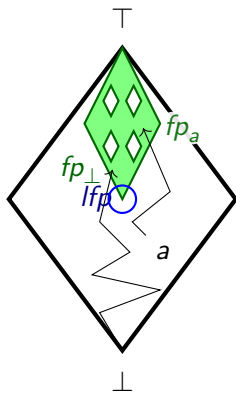
- ▶ Repeat $trans_P$ until fixpoint
- ▶ Can start from *any* point a
- ▶ Multiple **fixpoints** possible

Fixpoints



- ▶ Repeat $trans_P$ until fixpoint
 - ▶ Can start from *any* point a
 - ▶ Multiple **fixpoints** possible
 - ▶ Each is a *sound* solution
 - ▶ Form a lattice
- (Knaster-Tarski, 1933)

Fixpoints



- ▶ Repeat $trans_P$ until fixpoint
- ▶ Can start from *any* point a
- ▶ Multiple **fixpoints** possible
 - ▶ Each is a *sound* solution
 - ▶ 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
  }
}
```

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

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

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

- ▶ 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

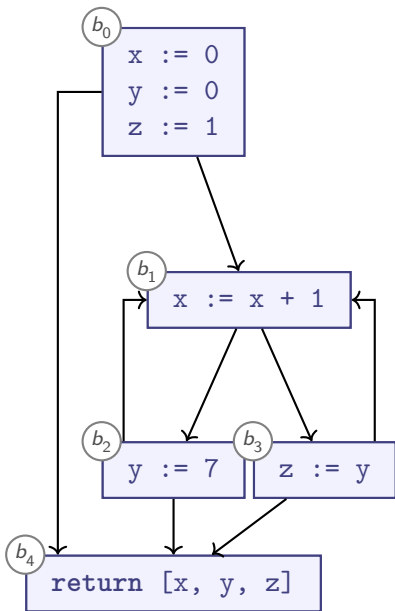
An Algorithm for Fixpoints

- ▶ 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*

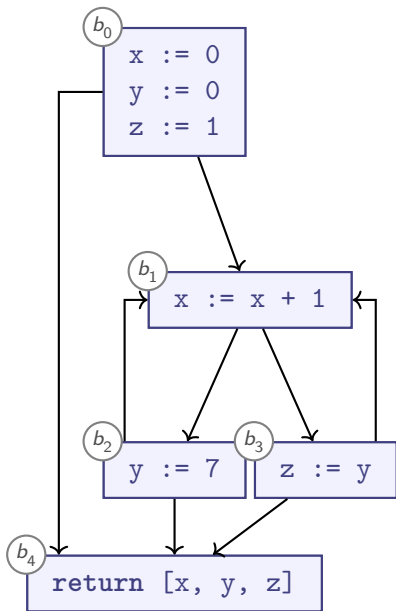
An Algorithm for Fixpoints

- ▶ 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*
- ⇒ “MFP” (Minimal Fixed Point) Algorithm
(*Does not always produce least fixpoint → will see later today*)

MFP Example:



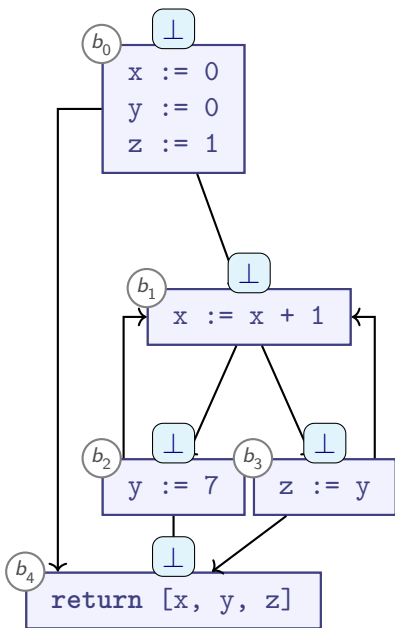
MFP Example:



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

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

MFP Example:



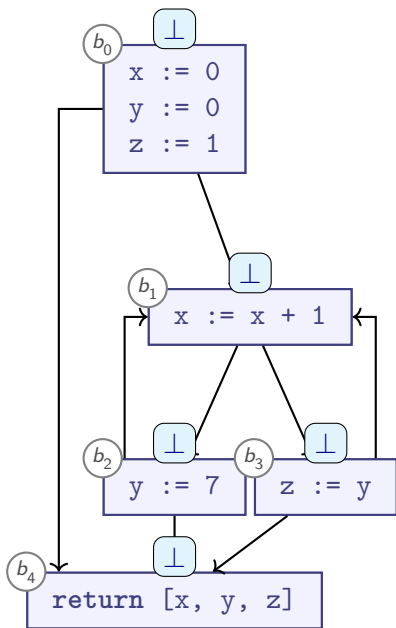
b	inputs	$trans_b$		
		x	y	z
b_0	\emptyset	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x + 1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

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

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

$b_0 \rightarrow b_1$

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

$b_1 \rightarrow b_3$

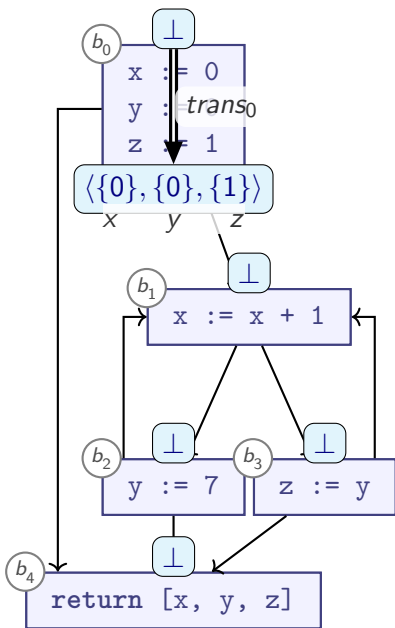
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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

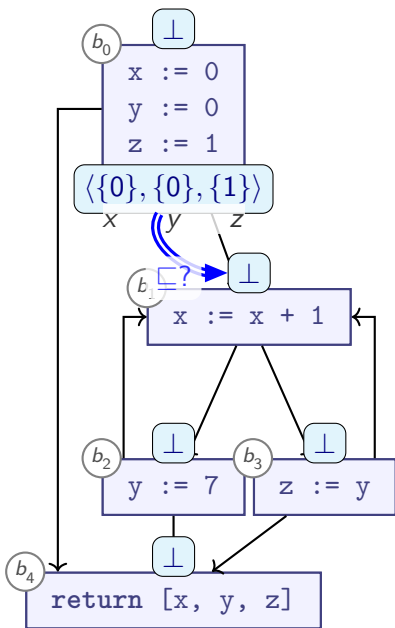
$$\begin{aligned}
 \text{join}_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1} \rangle, \langle v_{x_2}, v_{y_2}, v_{z_2} \rangle) = \\
 \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2} \rangle
 \end{aligned}$$

For edge $b_0 \rightarrow b_i$:

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 $out_o \sqsubseteq in_i$?

Worklist

$b_0 \rightarrow b_1$

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

$b_1 \rightarrow b_3$

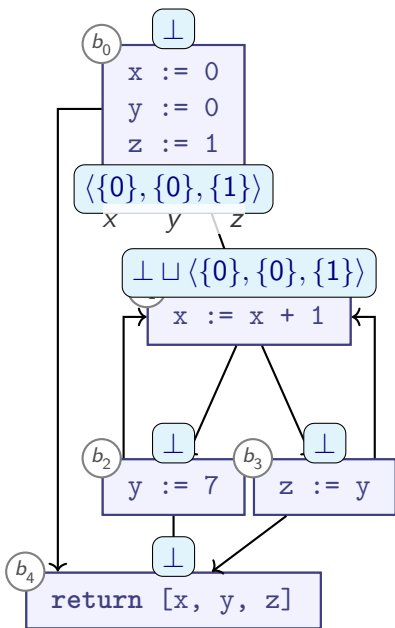
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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

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

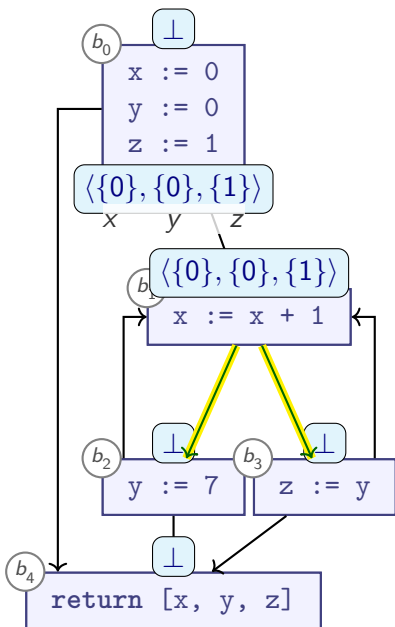
For edge $b_o \rightarrow b_i$:

- ▶ Is $out_o \sqsubseteq in_i$?
- ▶ Yes:
 - ▶ $in_i := in_i \sqcup out_o$

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	∅	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

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

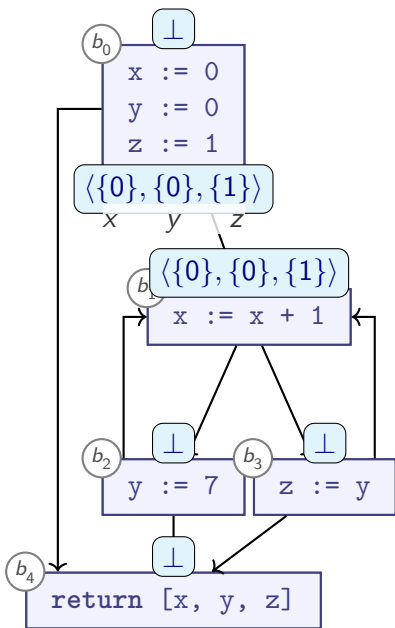
For edge $b_o \rightarrow b_i$:

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

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

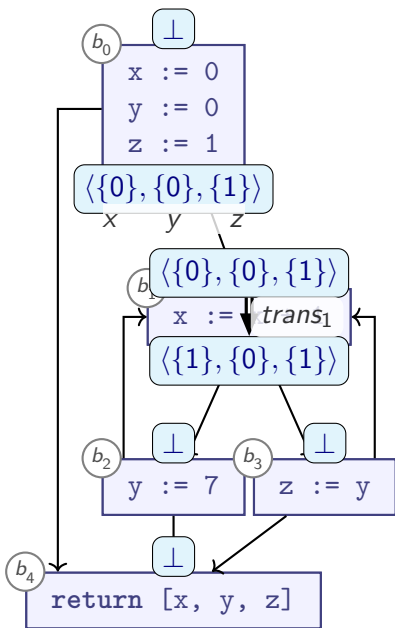
For edge $b_o \rightarrow b_i$:

- ▶ Is $\text{out}_o \sqsubseteq \text{in}_i$?
- ▶ Yes:
 - ▶ $\text{in}_i := \text{in}_i \sqcup \text{out}_o$
 - ▶ 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

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

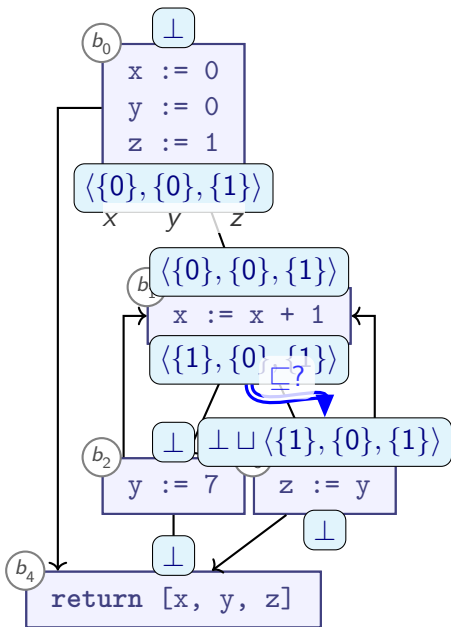
For edge $b_o \rightarrow b_i$:

- ▶ Is $out_o \sqsubseteq in_i$?
- ▶ Yes:
 - ▶ $in_i := in_i \sqcup out_o$
 - ▶ Add all outgoing edges from b_o to worklist (if not already there)

Worklist

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

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

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

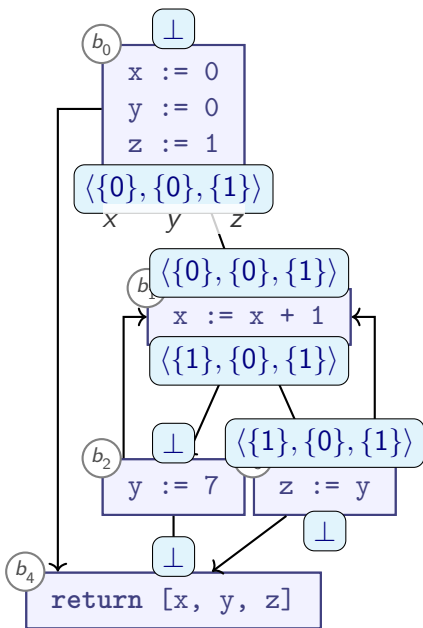
For edge $b_o \rightarrow b_i$:

- ▶ Is $out_o \not\sqsubseteq in_i$?
- ▶ Yes:
 - ▶ $in_i := in_i \sqcup out_o$
 - ▶ Add all outgoing edges from b_o to worklist (if not already there)

Worklist

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

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

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

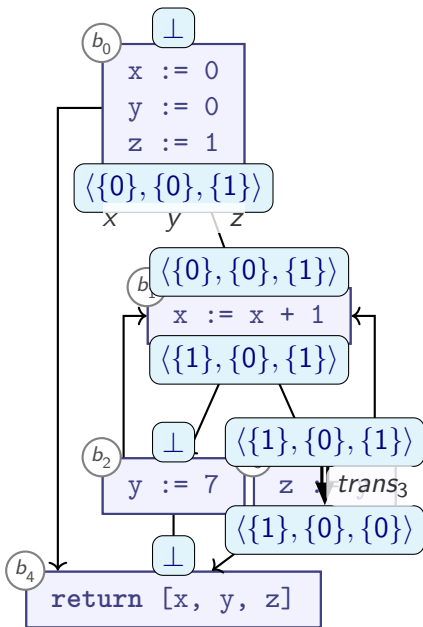
For edge $b_o \rightarrow b_i$:

- ▶ Is $out_o \not\sqsubseteq in_i$?
- ▶ Yes:
 - ▶ $in_i := in_i \sqcup out_o$
 - ▶ Add all outgoing edges from b_o to worklist (if not already there)

Worklist

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

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 $out_o \sqsubseteq in_i$?
- ▶ Yes:
 - ▶ $in_i := in_i \sqcup out_o$
 - ▶ Add all outgoing edges from b_o to worklist (if not already there)

Worklist

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

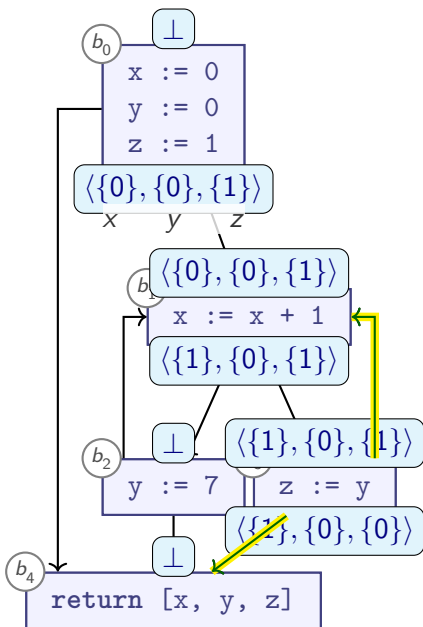
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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 $b_o \rightarrow b_i$:

- ▶ 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

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

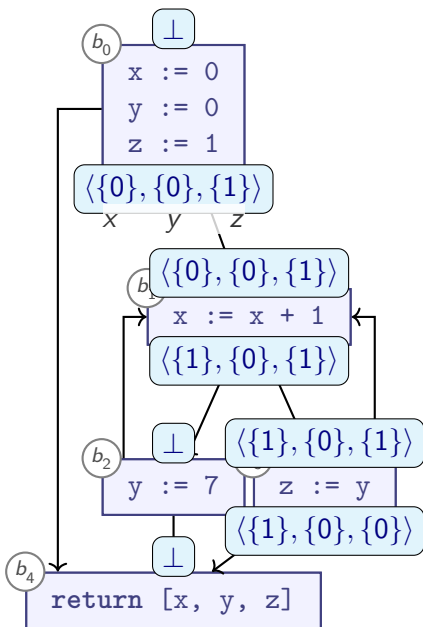
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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 $b_o \rightarrow b_i$:

- ▶ 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

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

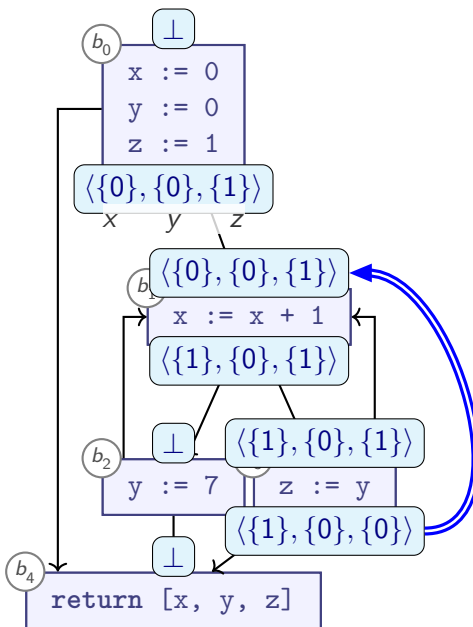
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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 $b_o \rightarrow b_i$:

- ▶ 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

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

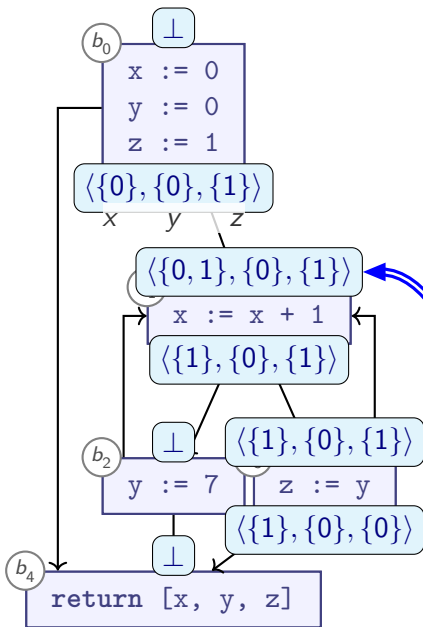
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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 $b_o \rightarrow b_i$:

- ▶ 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

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

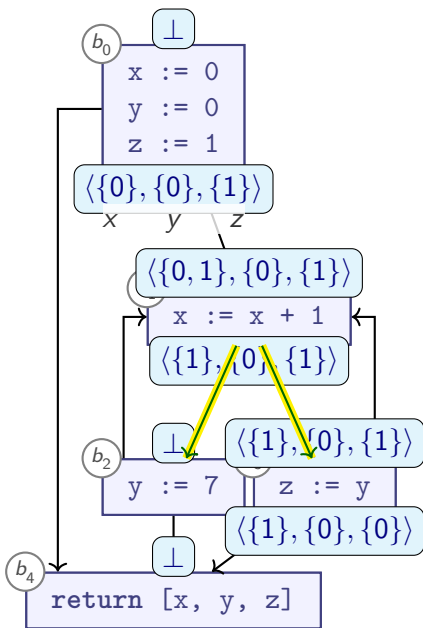
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

~~$b_2 \rightarrow b_1$~~

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 $out_o \not\sqsubseteq in_i$?
- ▶ Yes:
 - ▶ $in_i := in_i \sqcup out_o$
 - ▶ Add all outgoing edges from b_i to worklist (if not present)

Re-add previously removed edge

Worklist

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_1 \rightarrow b_3$

The MFP Algorithm

```
Procedure MFP( $\perp$ ,  $\sqcup$ ,  $\sqsubseteq$ , CFG, trans_, is-backward):  
begin  
  if is-backward then reverse edges(CFG);  
  worklist := edges(CFG); -- edges that we need to look at  
  foreach  $n \in \text{nodes}(\text{CFG})$  do  
    in[n] :=  $\perp$ ;           -- state of the analysis  
  done  
  while not empty(worklist) do  
     $\langle n, n' \rangle$  := pop(worklist); -- Edge  $n \rightarrow n'$   
    -- OPTIONAL: cache out[n] = transn(in[n]) here  
    if transn(in[n])  $\not\sqsubseteq$  in[n'] then begin  
      in[n'] := in[n']  $\sqcup$  transn(in[n]);  
      foreach  $n'' \in \text{successor-nodes}(\text{CFG}, n')$  do  
        push(worklist,  $\langle n', n'' \rangle$ );  
      done  
    end  
  done  
  return in;  
end
```

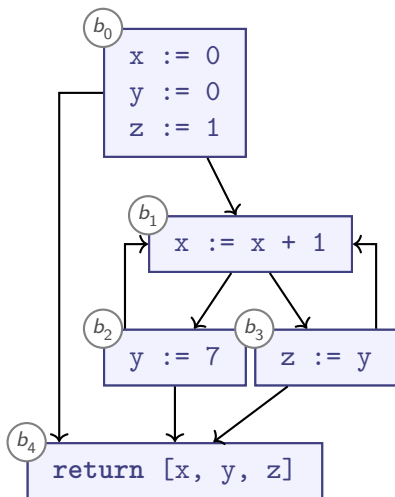
Worklist allows focussing effort!

Summary: MFP Algorithm

- ▶ **Product Lattice** allows analysing multiple variables at once
- ▶ Compute data flow analysis:
 - ▶ Initialise all nodes with \perp
 - ▶ 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

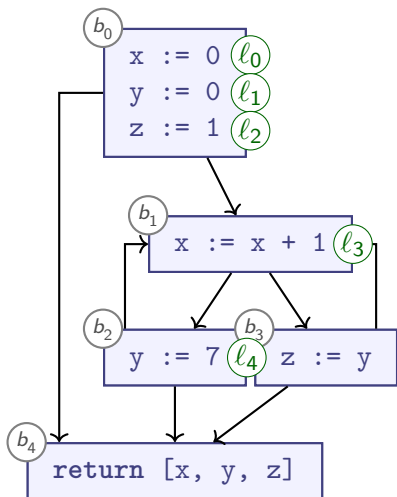
MFP revisited

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



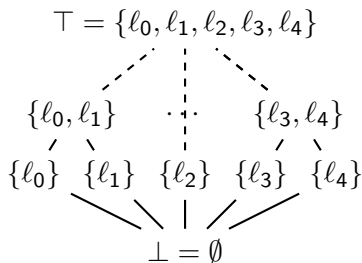
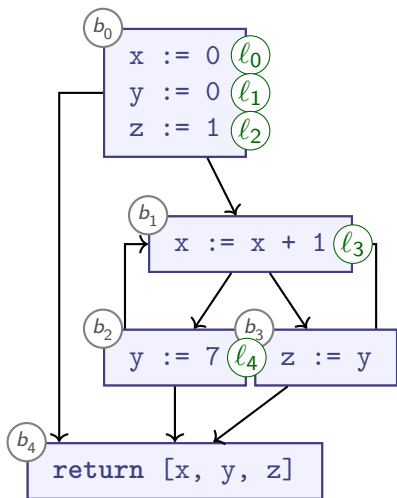
MFP revisited

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



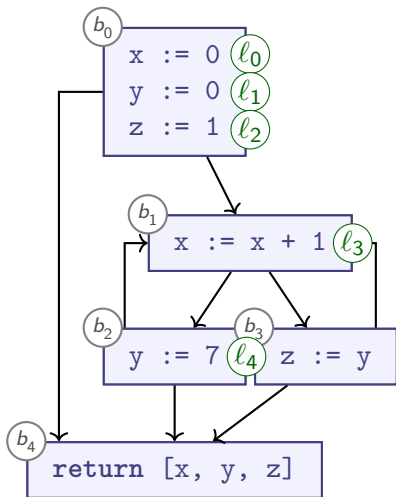
MFP revisited

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



- ▶ All subsets of $\{l_0, \dots, l_4\}$
- ▶ Finite height
- ▶ $\sqcup = \cup$

MFP revisited: Transfer Functions



$$trans_{b_0} = [x \mapsto \{l_0\}, \\ y \mapsto \{l_1\}, \\ z \mapsto \{l_2\}]$$

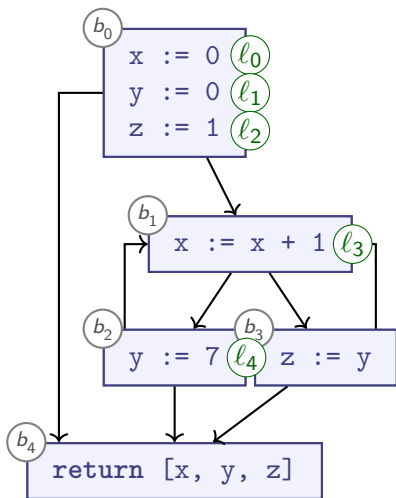
$$trans_{b_1} = [x \mapsto \{l_3\}]$$

$$trans_{b_2} = [y \mapsto \{l_4\}]$$

$$trans_{b_3} = [z \mapsto y]$$

MFP solution

MFP revisited: Transfer Functions



$$\text{trans}_{b_0} = [x \mapsto \{l_0\}, \\ y \mapsto \{l_1\}, \\ z \mapsto \{l_2\}]$$

$$\text{trans}_{b_1} = [x \mapsto \{l_3\}]$$

$$\text{trans}_{b_2} = [y \mapsto \{l_4\}]$$

$$\text{trans}_{b_3} = [z \mapsto y]$$

MFP solution

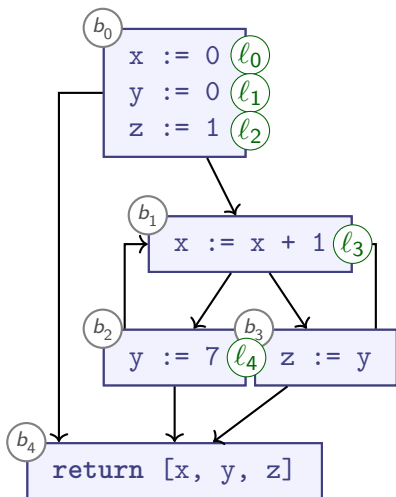
$$x \mapsto \{l_0, l_3\}$$

$$y \mapsto \{l_1, l_4\}$$

$$z \mapsto \{l_1, l_2, l_4\}$$

- Least Fixpoint!

MFP revisited: Transfer Functions



$$trans_{b_0} = [x \mapsto \{l_0\}, \\ y \mapsto \{l_1\}, \\ z \mapsto \{l_2\}]$$

$$trans_{b_1} = [x \mapsto \{l_3\}]$$

$$trans_{b_2} = [y \mapsto \{l_4\}]$$

$$trans_{b_3} = [z \mapsto y]$$

MFP solution

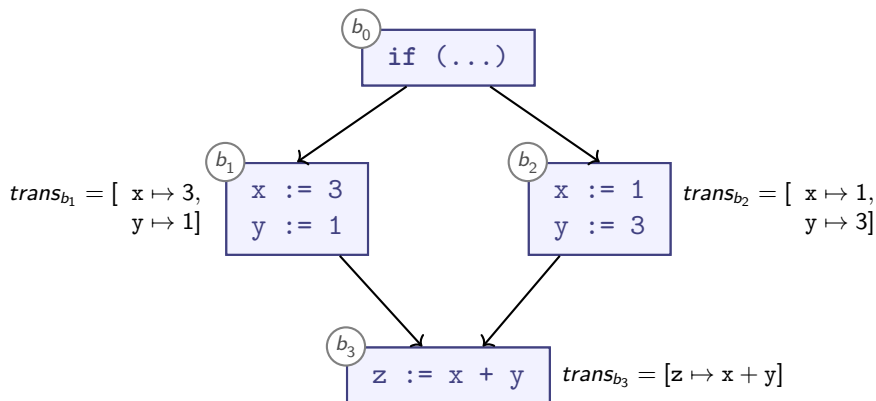
$$x \mapsto \{l_0, l_3\}$$

$$y \mapsto \{l_1, l_4\}$$

$$z \mapsto \{l_1, l_2, l_4\}$$

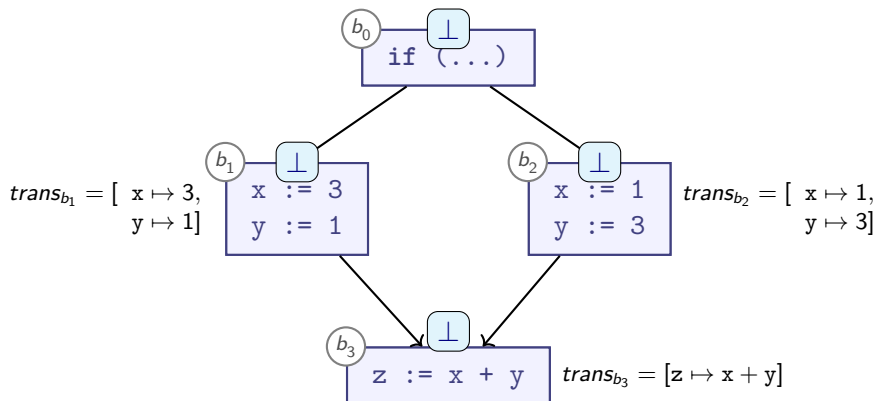
- ▶ Least Fixpoint!
- ▶ Do we always get LFP from MFP?

Another Example



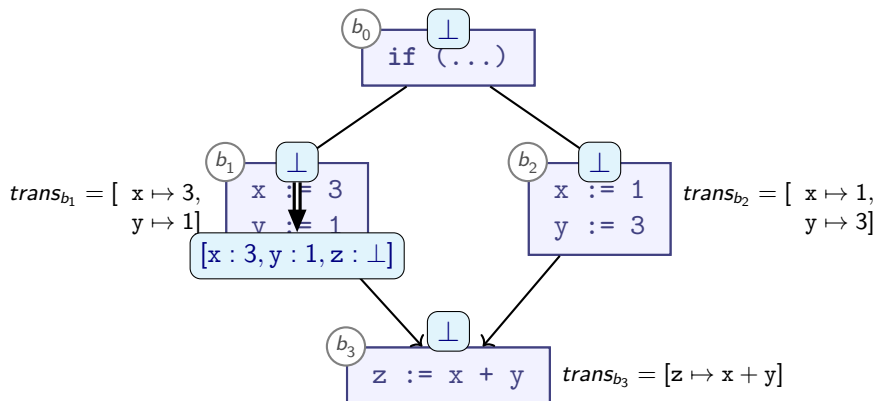
► Lattice: \mathbb{Z}_{\perp}^T

Another Example



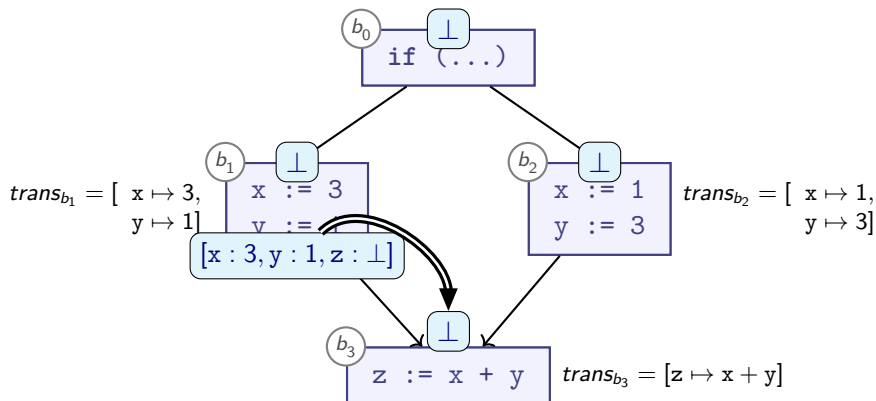
► Lattice: $\mathbb{Z}_{\perp}^{\top}$

Another Example



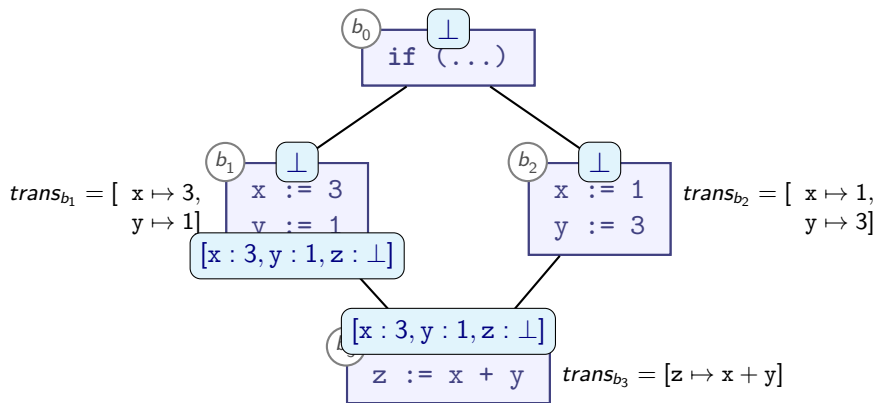
► Lattice: $\mathbb{Z}_{\perp}^{\top}$

Another Example



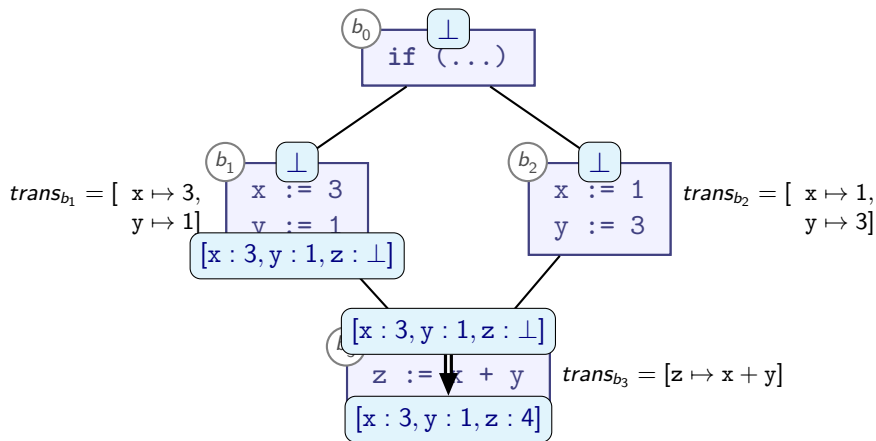
► Lattice: $\mathbb{Z}_{\perp}^{\top}$

Another Example



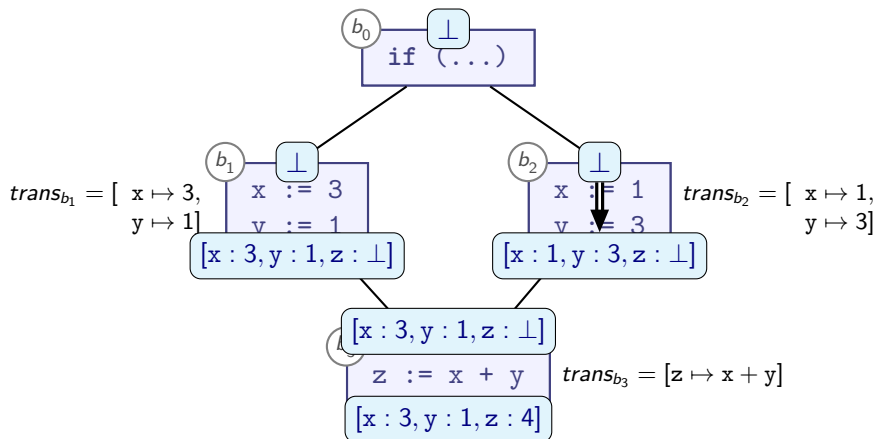
► Lattice: $\mathbb{Z}_{\perp}^{\top}$

Another Example



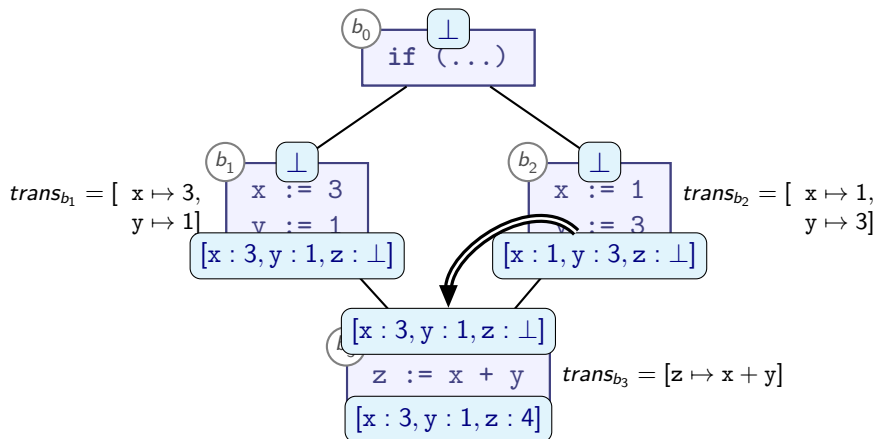
► Lattice: $\mathbb{Z}_{\perp}^{\top}$

Another Example



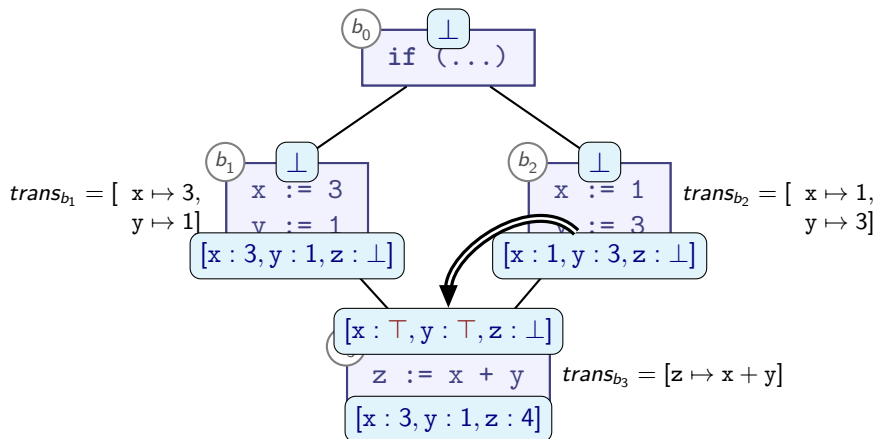
► Lattice: \mathbb{Z}_{\perp}^T

Another Example



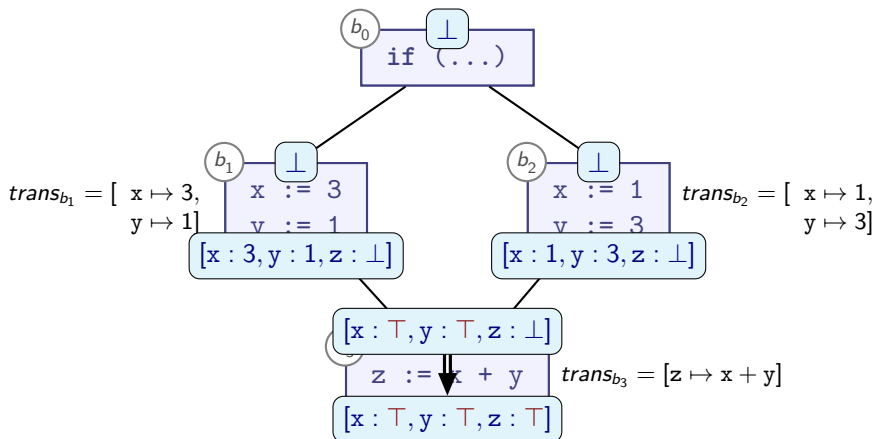
- ▶ Lattice: \mathbb{Z}_{\perp}^T
 - ▶ $1 \sqcup 3 = \top = 3 \sqcup 1$

Another Example



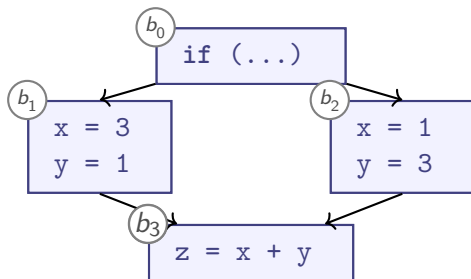
- ▶ Lattice: $\mathbb{Z}_{\perp}^{\top}$
 - ▶ $1 \sqcup 3 = \top = 3 \sqcup 1$

Another Example



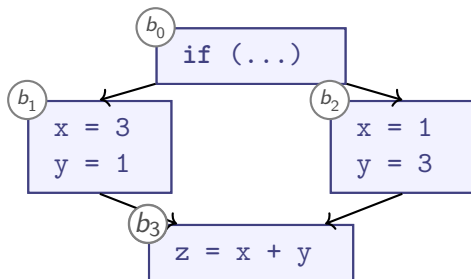
- ▶ Lattice: $\mathbb{Z}_{\perp}^{\top}$
 - ▶ $1 \sqcup 3 = \top = 3 \sqcup 1$
- ▶ MFP **does** compute the Least Fixpoint in our equations. . .
- ▶ . . . but the fixpoint is worse than expected!

Execution paths



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

Execution paths



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

$$path_{b_0} = \{[]\}$$

$$path_{b_1} = \{[b_0]\}$$

$$path_{b_2} = \{[b_0]\}$$

$$path_{b_3} = \{[b_0, b_1], [b_0, b_2]\}$$

The MOP algorithm for Dataflow Analysis

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

The MOP algorithm for Dataflow Analysis

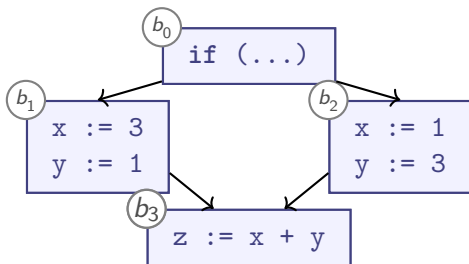
- ▶ Compute the MOP ('meet-over-all-paths') solution:
 - ▶ Iterate over all paths $[p_0, \dots, p_k]$ in $path_{b_i}$
 - ▶ Compute *precise* result for that path
 - ▶ Merge (i.e., join, \sqcup) with all other precise results

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

Notation: (*function composition*)

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

MOP vs MFP: Example



Transfer functions

$$trans_{b_0} = id$$

$$trans_{b_1} = [x \mapsto 3][y \mapsto 1]$$

$$trans_{b_2} = [x \mapsto 1][y \mapsto 3]$$

$$trans_{b_3} = [z \mapsto x + y]$$

Paths

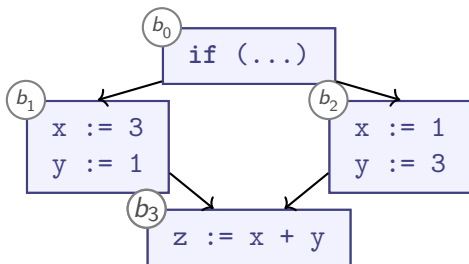
$$path_{b_0} = \{\emptyset\}$$

$$path_{b_1} = \{[b_0]\}$$

$$path_{b_2} = \{[b_0]\}$$

$$path_{b_3} = \{[b_0, b_1], [b_0, b_2]\}$$

MOP vs MFP: Example



Transfer functions

$$trans_{b_0} = id$$

$$trans_{b_1} = [x \mapsto 3][y \mapsto 1]$$

$$trans_{b_2} = [x \mapsto 1][y \mapsto 3]$$

$$trans_{b_3} = [z \mapsto x + y]$$

Paths

$$path_{b_0} = \{\emptyset\}$$

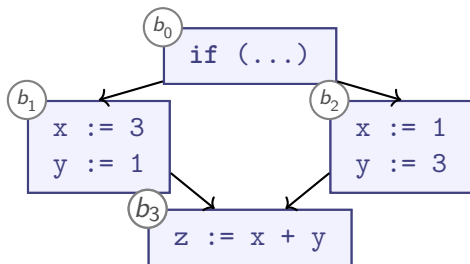
$$path_{b_1} = \{[b_0]\}$$

$$path_{b_2} = \{[b_0]\}$$

$$path_{b_3} = \{[b_0, b_1], [b_0, b_2]\}$$

$$\mathbf{out}_{b_3} = ([z \mapsto x + y][x \mapsto 3][y \mapsto 1](\perp)) \sqcup ([z \mapsto x + y][x \mapsto 1][y \mapsto 3](\perp))$$

MOP vs MFP: Example



Transfer functions

$$trans_{b_0} = id$$

$$trans_{b_1} = [x \mapsto 3][y \mapsto 1]$$

$$trans_{b_2} = [x \mapsto 1][y \mapsto 3]$$

$$trans_{b_3} = [z \mapsto x + y]$$

Paths

$$path_{b_0} = \{\emptyset\}$$

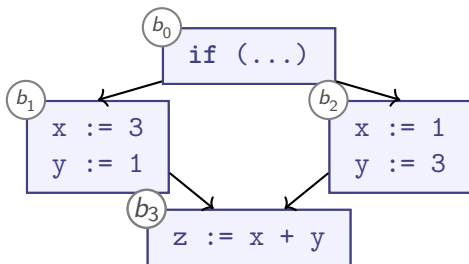
$$path_{b_1} = \{[b_0]\}$$

$$path_{b_2} = \{[b_0]\}$$

$$path_{b_3} = \{[b_0, b_1], [b_0, b_2]\}$$

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

MOP vs MFP: Example



Transfer functions

$$trans_{b_0} = id$$

$$trans_{b_1} = [x \mapsto 3][y \mapsto 1]$$

$$trans_{b_2} = [x \mapsto 1][y \mapsto 3]$$

$$trans_{b_3} = [z \mapsto x + y]$$

Paths

$$path_{b_0} = \{\emptyset\}$$

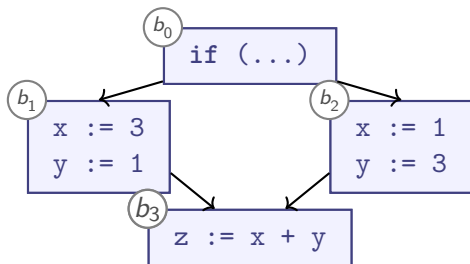
$$path_{b_1} = \{[b_0]\}$$

$$path_{b_2} = \{[b_0]\}$$

$$path_{b_3} = \{[b_0, b_1], [b_0, b_2]\}$$

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

MOP vs MFP: Example



Transfer functions

$$\text{trans}_{b_0} = \text{id}$$

$$\text{trans}_{b_1} = [x \mapsto 3][y \mapsto 1]$$

$$\text{trans}_{b_2} = [x \mapsto 1][y \mapsto 3]$$

$$\text{trans}_{b_3} = [z \mapsto x + y]$$

Paths

$$\text{path}_{b_0} = \{\emptyset\}$$

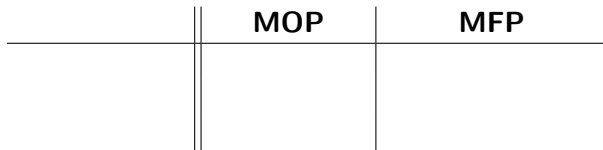
$$\text{path}_{b_1} = \{[b_0]\}$$

$$\text{path}_{b_2} = \{[b_0]\}$$

$$\text{path}_{b_3} = \{[b_0, b_1], [b_0, b_2]\}$$

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

MOP vs MFP



- ▶ MOP: Merge Over all Paths
(Originally: “Meet Over all Paths”, but we use the Join operator)
- ▶ MFP: Minimal Fixed Point

MOP vs MFP

	MOP	MFP
Soundness	sound	sound

- ▶ MOP: Merge Over all Paths
(Originally: “Meet Over all Paths”, but we use the Join operator)
- ▶ MFP: Minimal Fixed Point

MOP vs MFP

	MOP	MFP
Soundness	sound	sound
Precision	<i>maximal</i>	<i>sometimes lower</i>

- ▶ MOP: Merge Over all Paths
(Originally: “Meet Over all Paths”, but we use the Join operator)
- ▶ MFP: Minimal Fixed Point

MOP vs MFP

	MOP	MFP
Soundness	sound	sound
Precision	<i>maximal</i>	<i>sometimes lower</i>
Decidability	<i>undecidable</i>	<i>decidable</i>

- ▶ MOP: Merge Over all Paths
(Originally: “Meet Over all Paths”, but we use the Join operator)
- ▶ MFP: Minimal Fixed Point

Summary

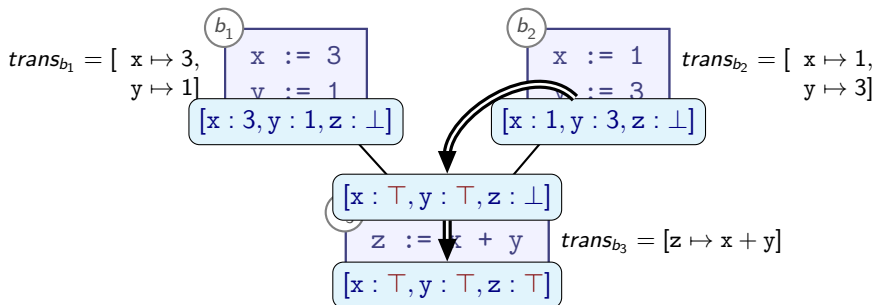
- ▶ $path_b$: Set of all paths from program start to b
- ▶ MOP: alternative to MFP (theoretically)
 - ▶ Termination not guaranteed
 - ▶ May be more precise
 - ▶ Idea:
 - ▶ Enumerate all paths to basic block
 - ▶ Compute transfer functions over paths individually
 - ▶ Join

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

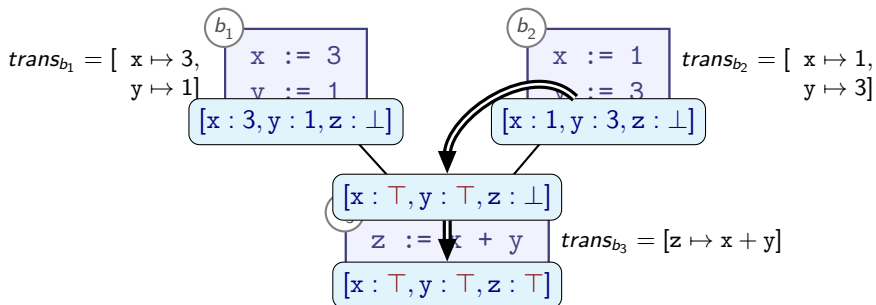
Why is MFP *sometimes* as good as MOP?

MFP vs the Least Fixpoint



- MFP is *sometimes* equal to MOP

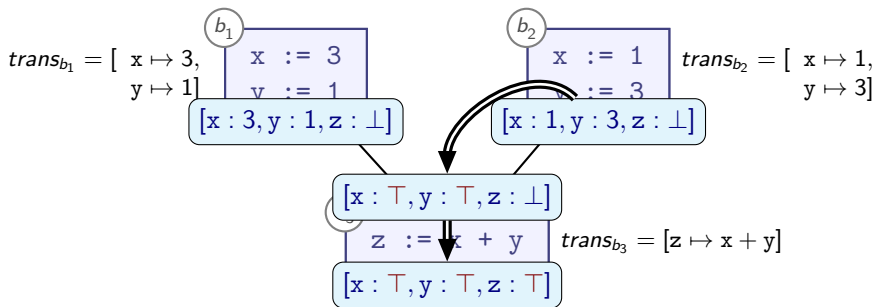
MFP vs the Least Fixpoint



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

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

MFP vs the Least Fixpoint



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

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

- ▶ **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

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

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:

$$\mathit{trans}_b(x \sqcup y) \sqsupseteq \mathit{trans}_b(x) \sqcup \mathit{trans}_b(y)$$

- ▶ Distributive:

$$\mathit{trans}_b(x \sqcup y) = \mathit{trans}_b(x) \sqcup \mathit{trans}_b(y)$$

Distributive Problems

- ▶ Monotonic:

$$\text{trans}_b(x \sqcup y) \sqsupseteq \text{trans}_b(x) \sqcup \text{trans}_b(y)$$

- ▶ Distributive:

$$\text{trans}_b(x \sqcup y) = \text{trans}_b(x) \sqcup \text{trans}_b(y)$$

- ▶ 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:

$$\text{trans}_b(x \sqcup y) = \text{trans}_b(x) \sqcup \text{trans}_b(y)$$

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