



LUND
UNIVERSITY

EDAP15: Program Analysis

DATAFLOW ANALYSIS 2
INTRAPROCEDURAL ANALYSIS

Christoph Reichenbach



Data Flow Analysis on CFGs

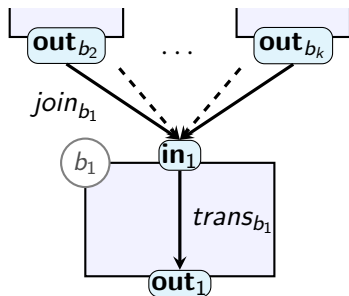
- ▶ $join_b$: Join Function
- ▶ $trans_b$: Transfer Function
- ▶ in_b : knowledge at entrance of b

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

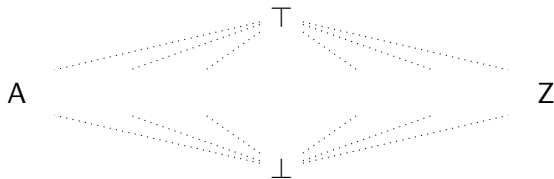
- ▶ out_b : knowledge at exit of b

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

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



Join and Transfer Functions



► L : Abstract Domain

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

$\top \in L$ 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

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

► pointwise monotonic

$$x \sqsubseteq y$$

$$\Downarrow$$

$$trans_b(x) \sqsubseteq trans_b(y)$$

$$x \sqsubseteq y$$

$$\Downarrow$$

$$join_b(z_1, \dots, z_k, x, \dots, z_n) \sqsubseteq join_b(z_1, \dots, z_k, y, \dots, z_n)$$

Partially Ordered Set

Lattices L are based on a *partially ordered set* $\langle \mathcal{L}, \sqsubseteq \rangle$:

- ▶ Set: \mathcal{L} describes possible information
- ▶ $(\sqsubseteq) \subseteq \mathcal{L} \times \mathcal{L}$:
- ▶ Intuition for $a \sqsubseteq b$ (for program analysis):
 - ▶ b has at least as much information as a
- ▶ (\sqsubseteq) is a *partial order*:

$$a \sqsubseteq a$$

Reflexivity

$$a \sqsubseteq b \text{ and } b \sqsubseteq a \implies a = b$$

Antisymmetry

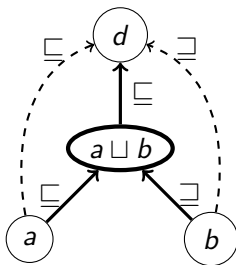
$$a \sqsubseteq b \text{ and } b \sqsubseteq c \implies a \sqsubseteq c$$

Transitivity

- ▶ Example:

- ▶ $\mathcal{L} = \{\text{unknown}, \text{true}, \text{false}, \text{true-or-false}\}$
- ▶ $\text{unknown} \sqsubseteq \text{true} \sqsubseteq \text{true-or-false}$
- ▶ $\text{unknown} \sqsubseteq \text{false} \sqsubseteq \text{true-or-false}$

Least Upper Bound



Combining potentially contradictory information:

- *Join operator*: $(\sqcup) : \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}$
- Pointwise monotonic:

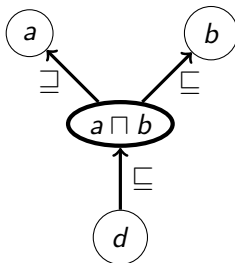
$$a \sqsubseteq a \sqcup b \text{ and } b \sqsubseteq a \sqcup b$$

- *Least* element with this property:

$$a \sqsubseteq d \text{ and } b \sqsubseteq d \implies a \sqcup b \sqsubseteq d$$

- '*Least*' ensures that this is the *most precise* model

Greatest Lower bound



Converse operation:

- *Meet operator*: $(\sqcap) : \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}$
- Pointwise monotonic:

$$a \sqcap b \sqsubseteq a \text{ and } a \sqcap b \sqsubseteq b$$

- *Greatest* element with this property:

$$d \sqsubseteq a \text{ and } d \sqsubseteq b \implies d \sqsubseteq a \sqcap b$$

Lattices

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

- ▶ \mathcal{L} : Underlying set
- ▶ $(\sqsubseteq) \subseteq \mathcal{L} \times \mathcal{L}$: Partial Order
- ▶ $(\sqcup) : \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}$: Join (computes l.u.b.)
- ▶ $(\sqcap) : \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{L}$: Meet (computes g.l.b.)
- ▶ Unique Join/Meet always exists
- ▶ We can show that (\sqcup) , (\sqcap) have:

$$\text{Commutativity:} \quad a \sqcup b = b \sqcup a$$

$$\text{Associativity:} \quad a \sqcup (b \sqcup c) = (a \sqcup b) \sqcup c$$

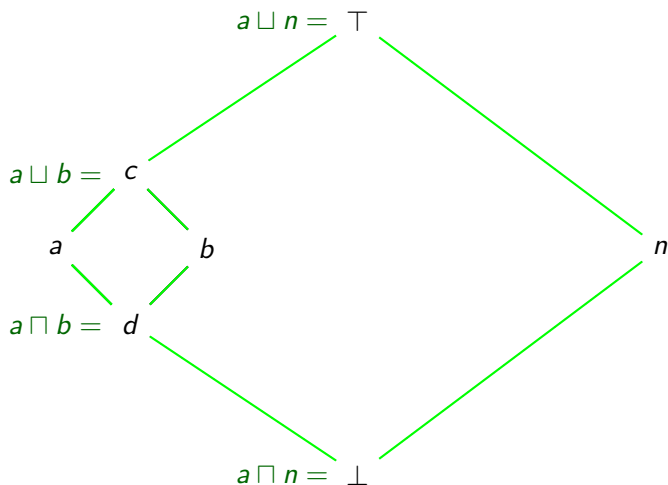
(Analogous for \sqcap)

Complete Lattices

A lattice $L = \langle \mathcal{L}, \sqsubseteq, \sqcap, \sqcup \rangle$ is *complete* iff:

- ▶ For any $\mathcal{L}' \subseteq \mathcal{L}$ there exist:
 - ▶ $\top = \sqcup \mathcal{L}'$ (least upper bound for arbitrary set)
 - ▶ $\perp = \sqcap \mathcal{L}'$ (greatest lower bound for arbitrary set)
- ▶ Usually trivial *in practice* (from a CS perspective)
 - ▶ Obvious if \mathcal{L} is finite
 - ▶ Counter-example: “Lattice of all finite subsets of \mathbb{N} ”

Complete Lattices: Visually



Example: Binary Lattice

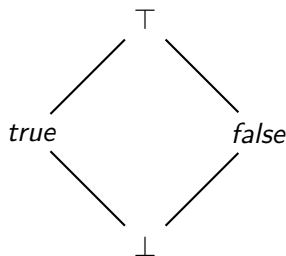
true



false

- ▶ $\top = \textit{true}$
- ▶ $\perp = \textit{false}$
- ▶ $\sqcup = \textit{logical "or"}$
- ▶ $\sqcap = \textit{logical "and"}$

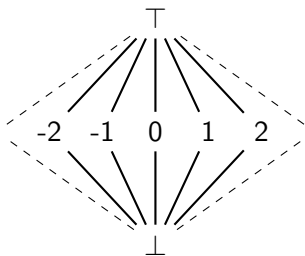
Example: Booleans



- ▶ If $\mathbb{B} = \{true, false\}$:
 - ▶ Lattice sometimes called $\mathbb{B}_{\perp}^{\top}$
- ▶ Interpretation for data flow e.g.:
 - ▶ \top = true-or-false
 - ▶ \perp = unknown
 - ▶ $a \sqcup b$: either a or b
 - ▶ $a \sqcap b$: both a and b

Other interpretations possible

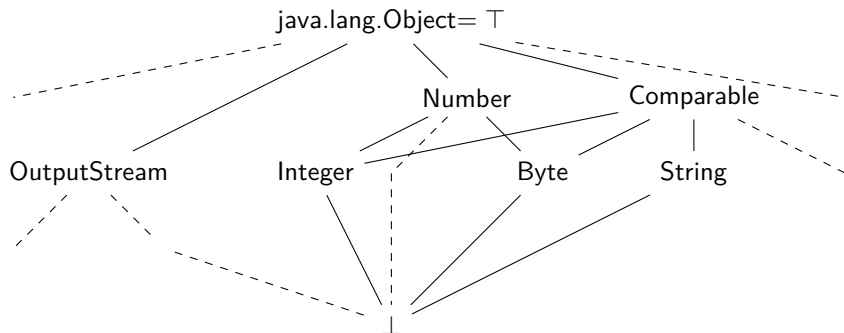
Example: Flat Lattice on Integers



- ▶ Sometimes written $\mathbb{Z}_{\perp}^{\top}$
- ▶ $\top = \emptyset$
- ▶ $\perp = \mathbb{Z}$
- ▶ $a \sqcup b = \begin{cases} a & \text{iff } a = b \\ \top & \text{otherwise} \end{cases}$
- ▶ $a \sqcap b = \begin{cases} a & \text{iff } a = b \\ \perp & \text{otherwise} \end{cases}$

Analogous for other X_{\perp}^{\top} from set X

Example: Type Hierarchy Lattices



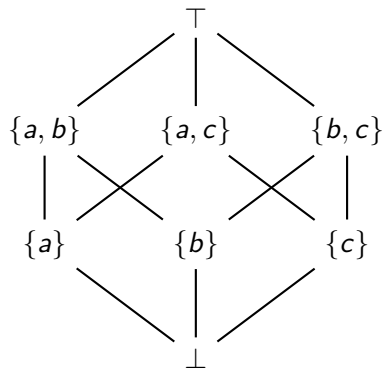
- ▶ \sqcup constructs most precise supertype
- ▶ \sqcap constructs *intersection types*:

`java.lang.Comparable` \sqcap `java.io.Serializable`

- ▶ *Java notation*:

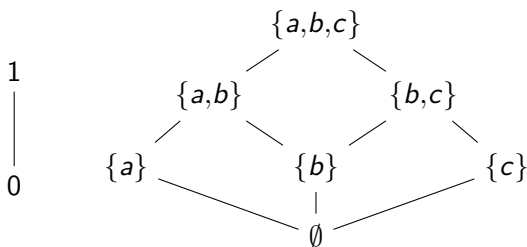
`java.lang.Comparable & java.io.Serializable`

Example: Powersets



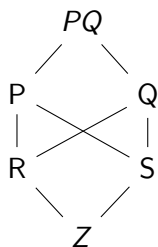
- ▶ Take any set $S = \{a, b, c\}$
- ▶ $\mathcal{L} = \mathcal{P}(S)$
- ▶ $\top = S$
- ▶ $\perp = \emptyset$
- ▶ $(\sqcup) = (\cup)$
- ▶ $(\sqcap) = (\cap)$

Example: Lattices and Non-Lattices



Lattice

Lattice



Not A Lattice

Right-hand side is missing e.g. a unique $R \sqcup S$

Example: Natural numbers with $0, \omega$

ω
⋮
3
|
2
|
1
|
0

- ▶ $\top = \omega$
- ▶ $\perp = 0$
- ▶ $a \sqcup b = \text{maximum of } a \text{ and } b$
- ▶ $a \sqcap b = \text{minimum of } a \text{ and } b$

Product Lattices

- ▶ Assume (complete) lattices:
 - ▶ $L_1 = \langle \mathcal{L}_1, \sqsubseteq_1, \sqcap_1, \sqcup_1, \top_1, \perp_1 \rangle$
 - ▶ $L_2 = \langle \mathcal{L}_2, \sqsubseteq_2, \sqcap_2, \sqcup_2, \top_2, \perp_2 \rangle$
- ▶ Let $L_1 \times L_2 = \langle \mathcal{L}_1 \times \mathcal{L}_2, \sqsubseteq, \sqcap, \sqcup, \top, \perp \rangle$ where:
 - ▶ $\langle a, b \rangle \sqsubseteq \langle a', b' \rangle$ iff $a \sqsubseteq_1 a'$ and $b \sqsubseteq_2 b'$
 - ▶ $\langle a, b \rangle \sqcap \langle a', b' \rangle = \langle a \sqcap_1 a', b \sqcap_2 b' \rangle$
 - ▶ $\langle a, b \rangle \sqcup \langle a', b' \rangle = \langle a \sqcup_1 a', b \sqcup_2 b' \rangle$
 - ▶ $\top = \langle \top_1, \top_2 \rangle$
 - ▶ $\perp = \langle \perp_1, \perp_2 \rangle$

Point-wise products of (complete) lattices are again (complete) lattices

Summary

- ▶ Complete lattices are formal basis for many program analyses
- ▶ Complete lattice $L = \langle \mathcal{L}, \sqsubseteq, \sqcap, \sqcup, \top, \perp \rangle$
 - ▶ \mathcal{L} : Carrier set
 - ▶ (\sqsubseteq) : Partial order
 - ▶ (\sqcap) : Join operation: find least upper lower bound
 - ▶ (\sqcup) : Meet operation: find greatest lower bound (not usually necessary)
 - ▶ \top : Top-most element of complete lattice
 - ▶ \perp : Bottom-most element of complete lattice
- ▶ **Product Lattices:** $L_1 \times L_2$ forms a lattice if L_1 and L_2 are lattices

Monotone Frameworks

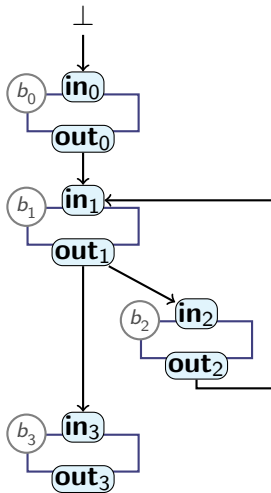
Monotone Framework	Lattice
Abstract Domain	$L = \langle \mathcal{L}, \sqsubseteq, \sqcap, \sqcup \rangle$
$join_b(x_1, \dots, x_n)$	$x_1 \sqcup \dots \sqcup x_n$
	$x \sqcap y$ (Not needed)
'Unknown' start value	\perp
'Could be anything' end value	\top
► <i>Monotone Frameworks</i> (Killdall '77):	
► Lattice L of <i>finite height</i> (= satisfies Ascending Chain Condition)	
► Monotone $trans_b$	
► 'compatible' with semantics	
⇒ Data flow analysis with Soundness and Termination	
► Don't need \sqcap , so technically we only need a <i>Semilattice</i> .	

Formalising our Naïve Algorithm

$$\begin{aligned}\mathbf{out}_0 &= trans_0(\perp) \\ \mathbf{out}_1 &= trans_1(\mathbf{out}_0 \sqcup \mathbf{out}_2) \\ \mathbf{out}_2 &= trans_2(\mathbf{out}_1) \\ \mathbf{out}_3 &= trans_3(\mathbf{out}_1)\end{aligned}$$

- ▶ Lattices $\mathbf{out}_0 : L_0, \dots, \mathbf{out}_3 : L_3$
- ▶ Can build lattice for entire program:
 - ▶ $L_{0\dots3} = L_0 \times L_1 \times L_2 \times L_3$
 - ▶ $\perp_{0\dots3} = \langle \perp_0, \perp_1, \perp_2, \perp_3 \rangle$
 - ▶ Monotone transfer function:

$$trans_{0\dots3}(\langle v_0, v_1, v_2, v_3 \rangle) = \left\langle \begin{array}{l} trans_0(v_0), \\ trans_1(v_0 \sqcup v_2), \\ trans_2(v_1), \\ trans_3(v_1) \end{array} \right\rangle$$



Reaching a Solution

- ▶ In general:
 - ▶ Program P :
 - ▶ “Program Lattice” L_P
 - ▶ \perp_P : initial analysis state
 - ▶ $trans_P$: Compute one step of naïve analysis
 - ▶ Repeat $trans_P$ until solution fp_{\perp} :

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

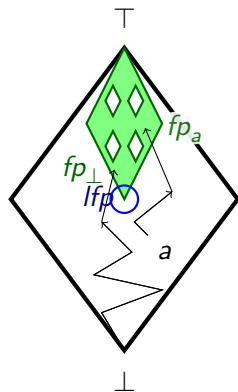
- ▶ n is the minimum number of steps until we have a solution
- ▶ fp_{\perp} is *Fixpoint* of $trans_P$:

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

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

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

Fixpoints



- ▶ 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 ’

Python

```
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*)

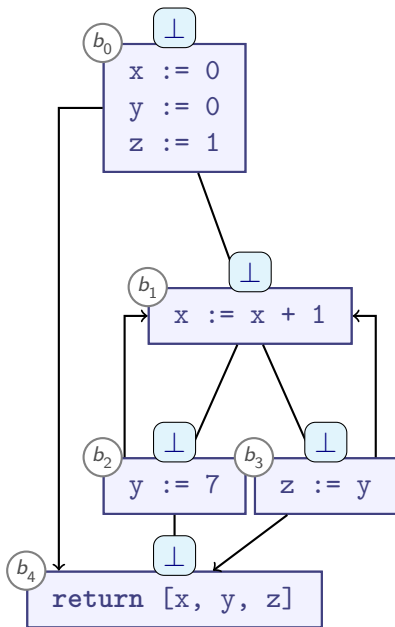
- ▶ Works from *any* starting point, possibly different fixpoint
 - ▶ Fixpoints form **Fixpoint Lattice**
 - ▶ **Least Fixpoint** (Bottom element) is *most precise solution*

- ▶ (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
 - ▶ 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!*)

MFP Example:



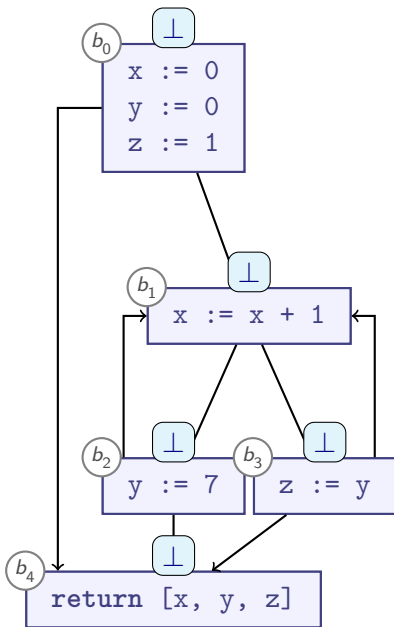
b	inputs	$trans_b$		
		x	y	z
b_0	\emptyset	0	0	1
b_1	$\{b_0, b_2, b_3\}$	$x + 1$	y	z
b_2	$\{b_1\}$	x	7	z
b_3	$\{b_1\}$	x	y	y
b_4	$\{b_0, b_2, b_3\}$	x	y	z

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

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

$b_0 \rightarrow b_1$

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

$b_1 \rightarrow b_3$

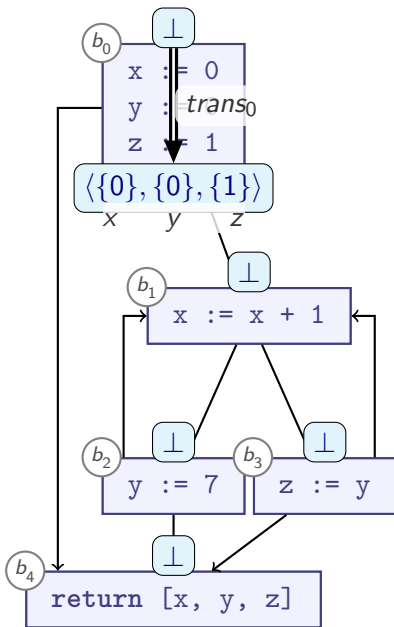
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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

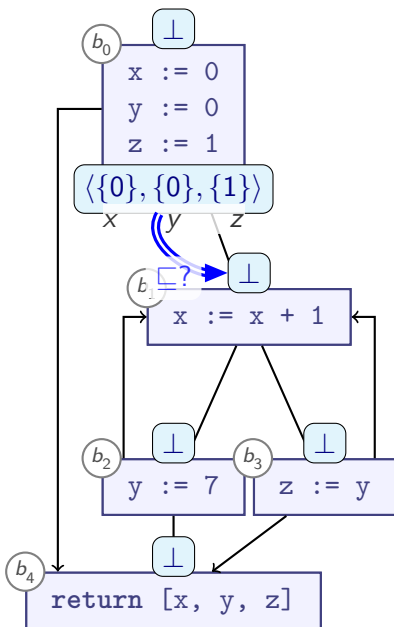
$$join_{b_i}(\langle v_{x_1}, v_{y_1}, v_{z_1} \rangle, \langle v_{x_2}, v_{y_2}, v_{z_2} \rangle) = \langle v_{x_1} \cup v_{x_2}, v_{y_1} \cup v_{y_2}, v_{z_1} \cup v_{z_2} \rangle$$

For edge $b_o \rightarrow b_i$:

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

$b_0 \rightarrow b_1$

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

$b_1 \rightarrow b_3$

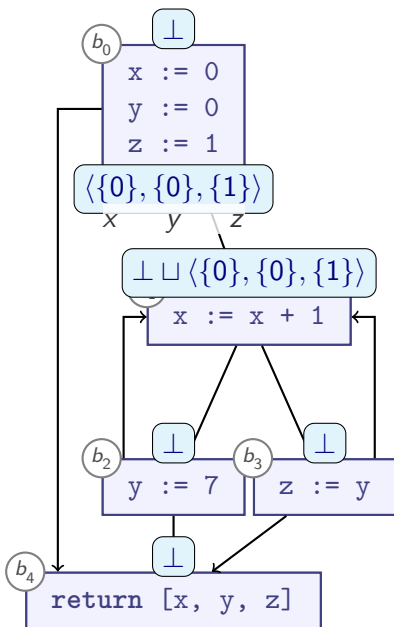
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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

$b_0 \rightarrow b_1$

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

$b_1 \rightarrow b_3$

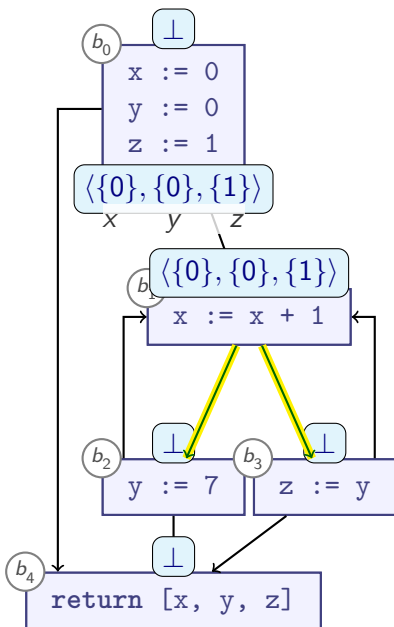
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

MFP Example:



b	inputs	$trans_b$		
		x	y	z
b_0	\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

$b_0 \rightarrow b_1$

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

$b_1 \rightarrow b_3$

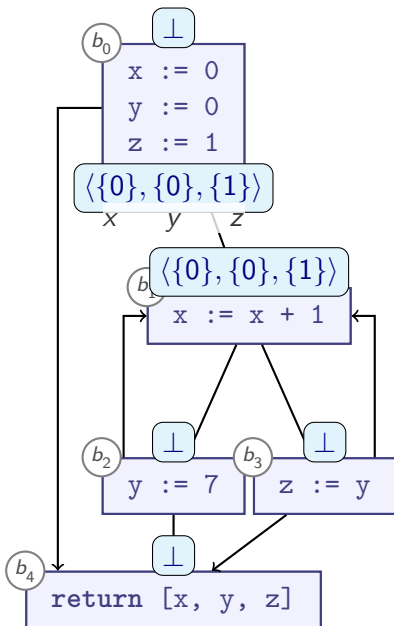
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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

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

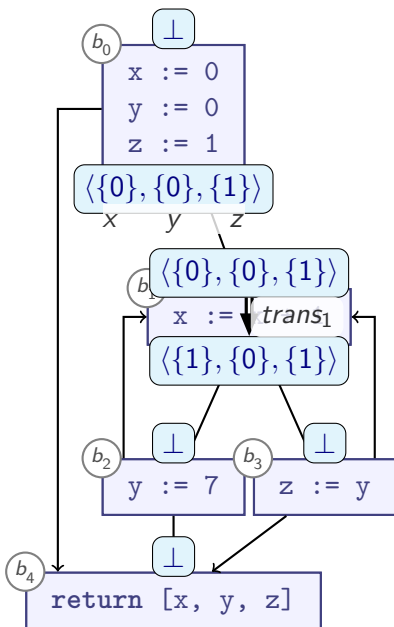
For edge $b_o \rightarrow b_i$:

- 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

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

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

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

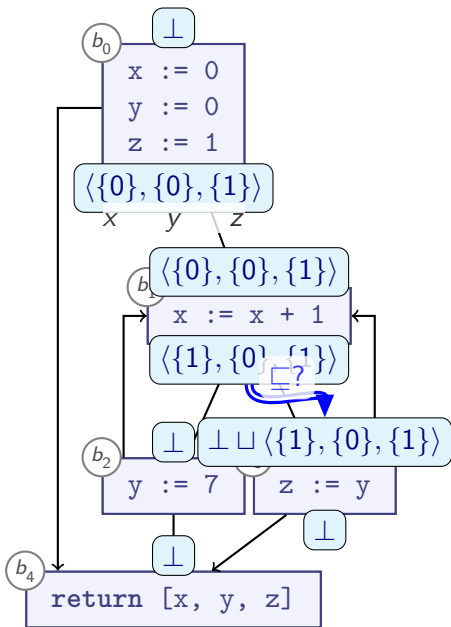
For edge $b_o \rightarrow b_i$:

- 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

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

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

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

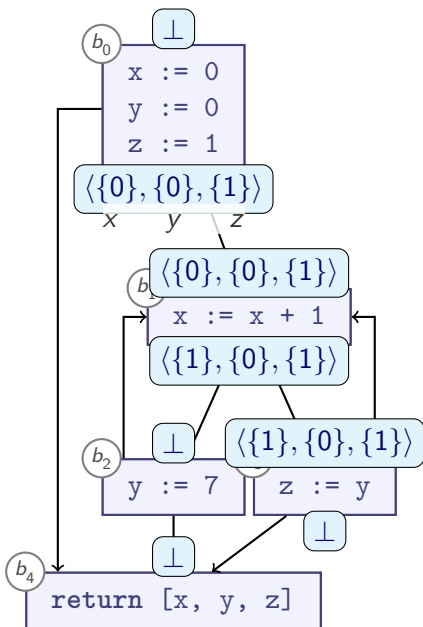
For edge $b_o \rightarrow b_i$:

- 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

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

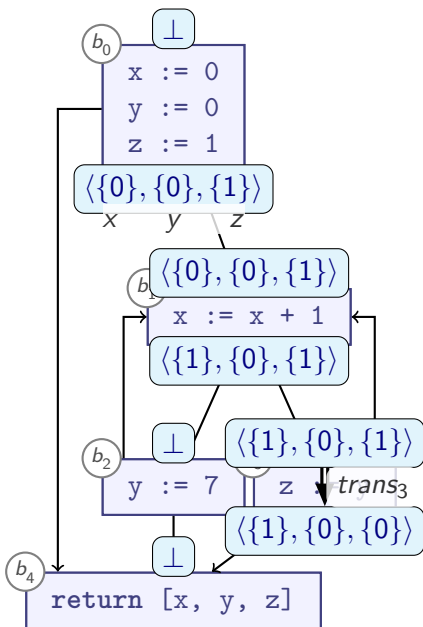
For edge $b_o \rightarrow b_i$:

- Is $out_o \sqsubseteq in_i$?
- 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

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

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

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

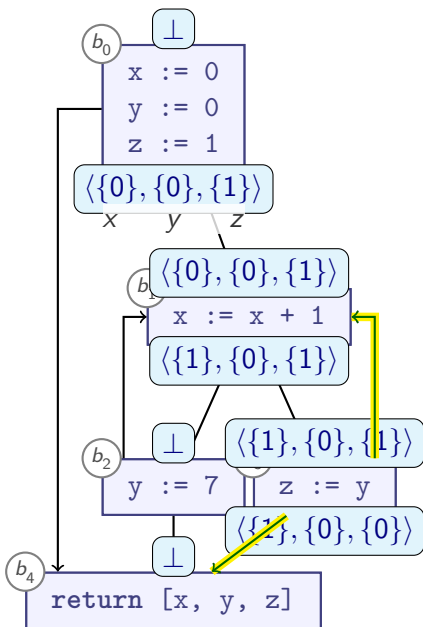
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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

Worklist

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

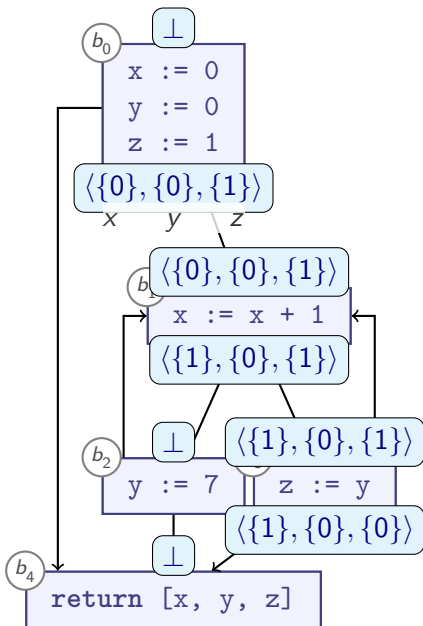
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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 $\text{out}_o \sqsubseteq \text{in}_i$?
- 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

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

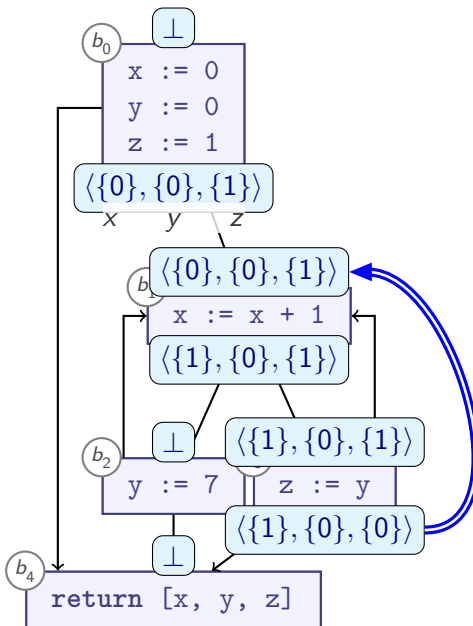
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

$b_3 \rightarrow b_1$

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 $b_o \rightarrow b_i$:

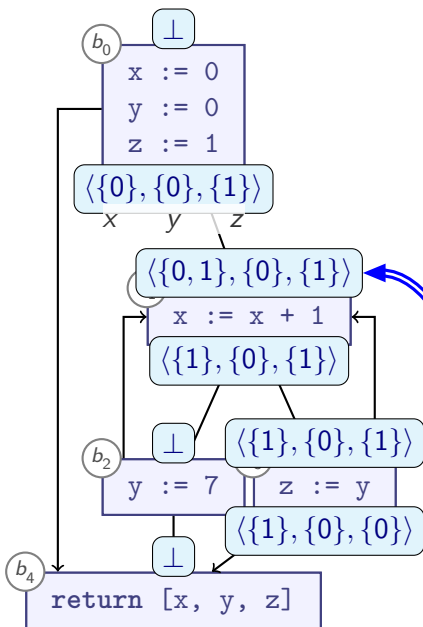
- Is $\text{out}_o \not\sqsubseteq \text{in}_i$?
- Yes:
 - $\text{in}_i := \text{in}_i \sqcup \text{out}_o$
 - Add all outgoing edges from b_o to worklist (if not already there)

Worklist

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

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

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

$b_0 \rightarrow b_4$

$b_1 \rightarrow b_2$

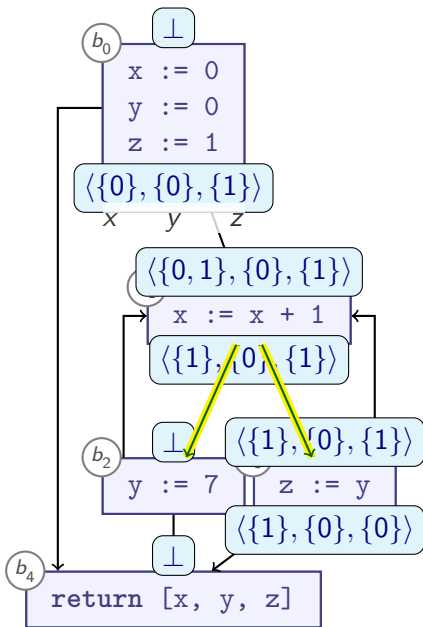
$b_2 \rightarrow b_4$

$b_2 \rightarrow b_1$

$b_3 \rightarrow b_4$

~~$b_2 \rightarrow b_1$~~

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 $out_o \sqsubseteq in_i$?

► Yes:

► $in_i := in_i \sqcup out_o$

► Add all outgoing edges from b_i to worklist (if not already 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
```

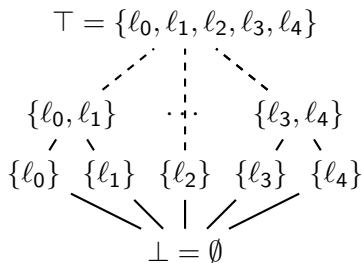
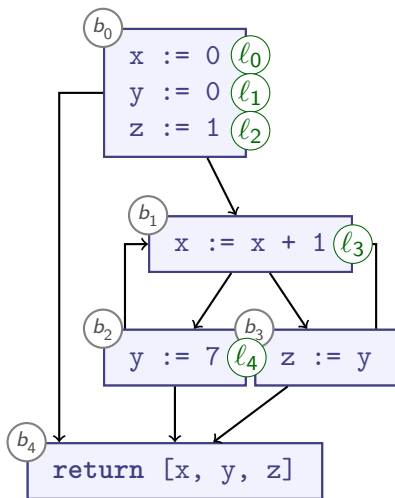
Worklist allows focussing effort!

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

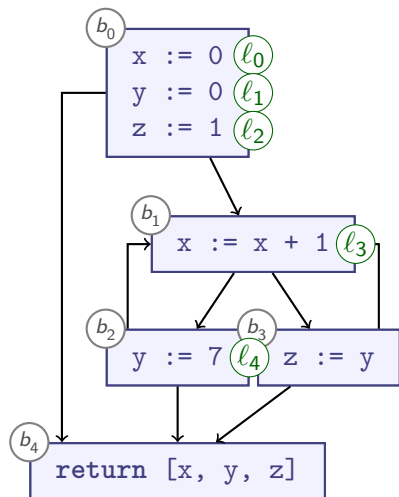
MFP revisited

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



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

MFP revisited: Transfer Functions



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

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

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

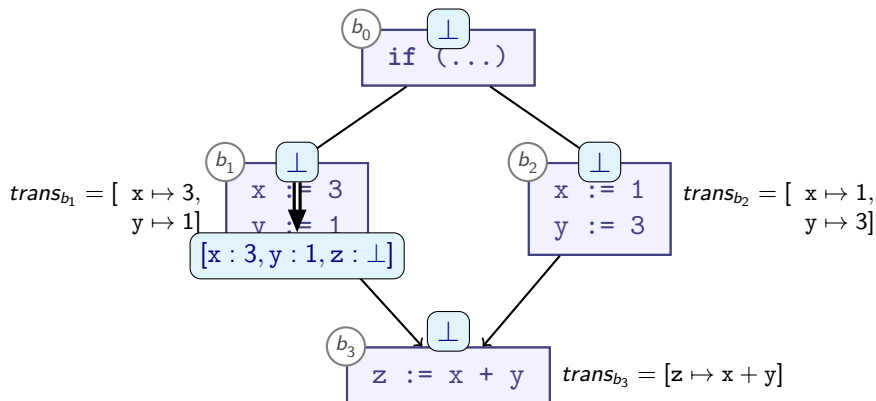
$$\text{trans}_{b_3} = [z \mapsto y]$$

MFP solution

$$\begin{array}{lcl} x & \mapsto & \{l_0, l_3\} \\ y & \mapsto & \{l_1, l_4\} \\ z & \mapsto & \{l_1, l_2, l_4\} \end{array}$$

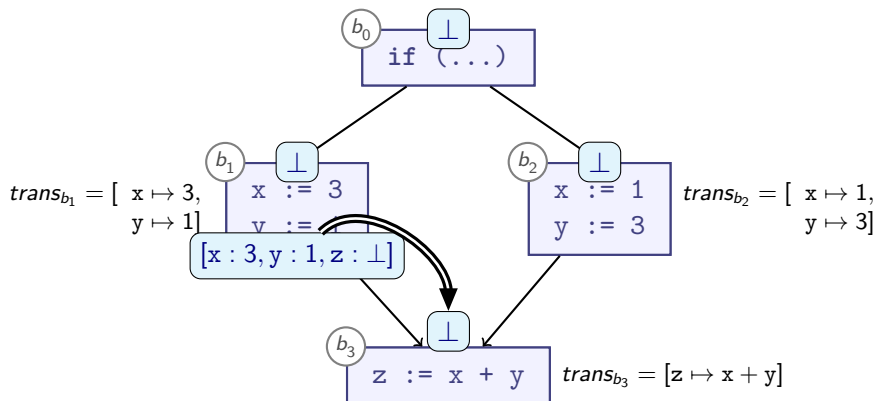
- ▶ Least Fixpoint!
- ▶ Do we always get LFP from MFP?

Another Example



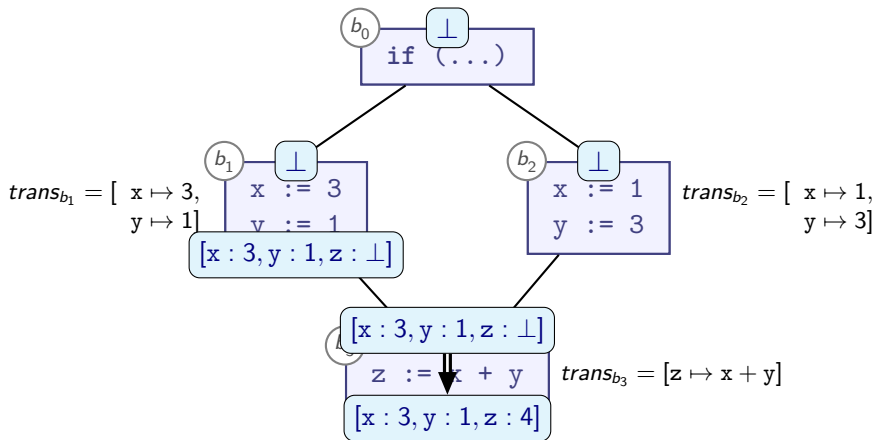
► Lattice: \mathbb{Z}_{\perp}^T

Another Example



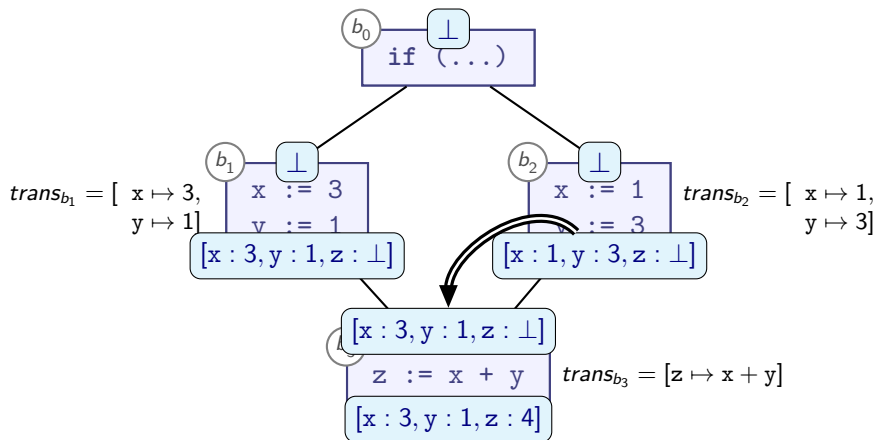
► Lattice: \mathbb{Z}_{\perp}^T

Another Example



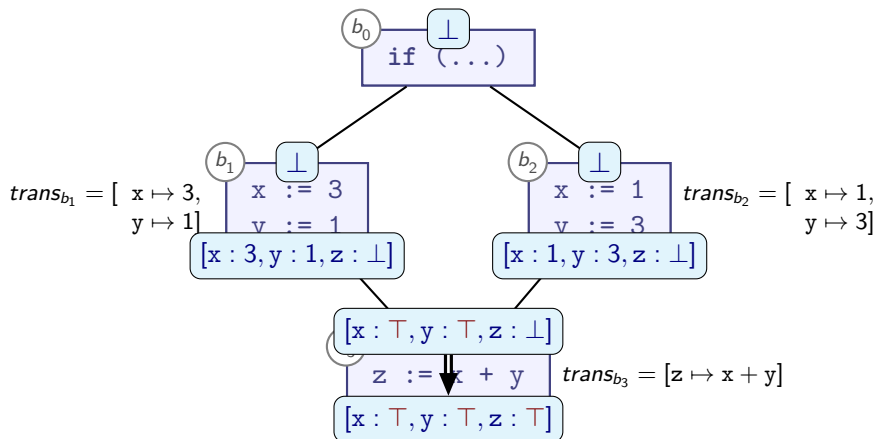
► Lattice: \mathbb{Z}_{\perp}^T

Another Example



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

Another Example



- Lattice: $\mathbb{Z}_{\perp}^{\top}$
 - $1 \sqcup 3 = \top = 3 \sqcup 1$
- **No**, MFP does not always compute the Least 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 Least Fixpoint

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 Least Fixpoint
- ▶ Some analyses (Gen/Kill analyses, discussed later) are always distributive