

# ABSTRACT FORWARD PROPAGATE

## KILDALL'S ALGORITHM

AbstractForwardPropagate( $\Gamma_c, P$ )

```
S := {l0};  
 $\hat{\mu}(l_0) := \alpha(P)$ ;  
 $\hat{\mu}(l) := \perp$ , for  $l \in L \setminus \{l_0\}$ ;  
while S  $\neq \emptyset$  do{  
    l := Choose S;  
    S := S  $\setminus$  {l};  
    foreach (l, c, l')  $\in T$  do{  
        F :=  $\hat{sp}(\hat{\mu}(l), c)$ ;  
        if  $\neg(F \leq \hat{\mu}(l'))$  then{  
             $\hat{\mu}(l') := \hat{\mu}(l') \sqcup F$ ;  
            S := S  $\cup$  {l'};  
        }  
    }  
}
```

- Does this algorithm actually calculate the abstract JOP?
- What are the conditions under which it is guaranteed to terminate?



# ABSTRACT FORWARD PROPAGATE

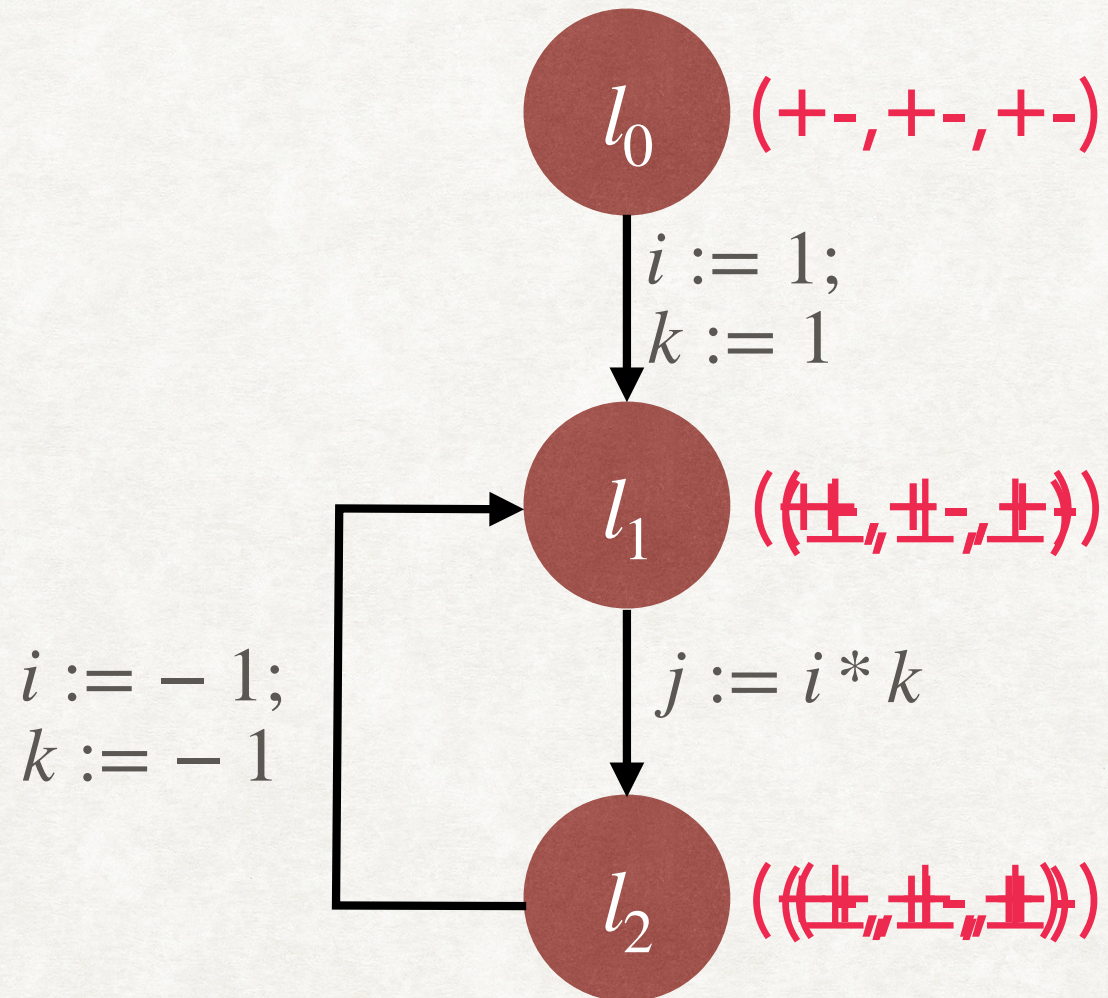
## KILDALL'S ALGORITHM

AbstractForwardPropagate( $\Gamma_c, P$ )

```
S := {l0};  
 $\hat{\mu}_K(l_0) := \alpha(P)$ ;  
 $\hat{\mu}_K(l) := \perp$ , for  $l \in L \setminus \{l_0\}$ ;  
while S  $\neq \emptyset$  do{  
    l := Choose S;  
    S := S  $\setminus$  {l};  
    foreach (l, c, l')  $\in T$  do{  
        F :=  $\hat{f}_c(\hat{\mu}_K(l))$ ;  
        if  $\neg(F \leq \hat{\mu}_K(l'))$  then{  
             $\hat{\mu}_K(l') := \hat{\mu}_K(l') \sqcup F$ ;  
            S := S  $\cup$  {l'};  
        }  
    }  
}
```

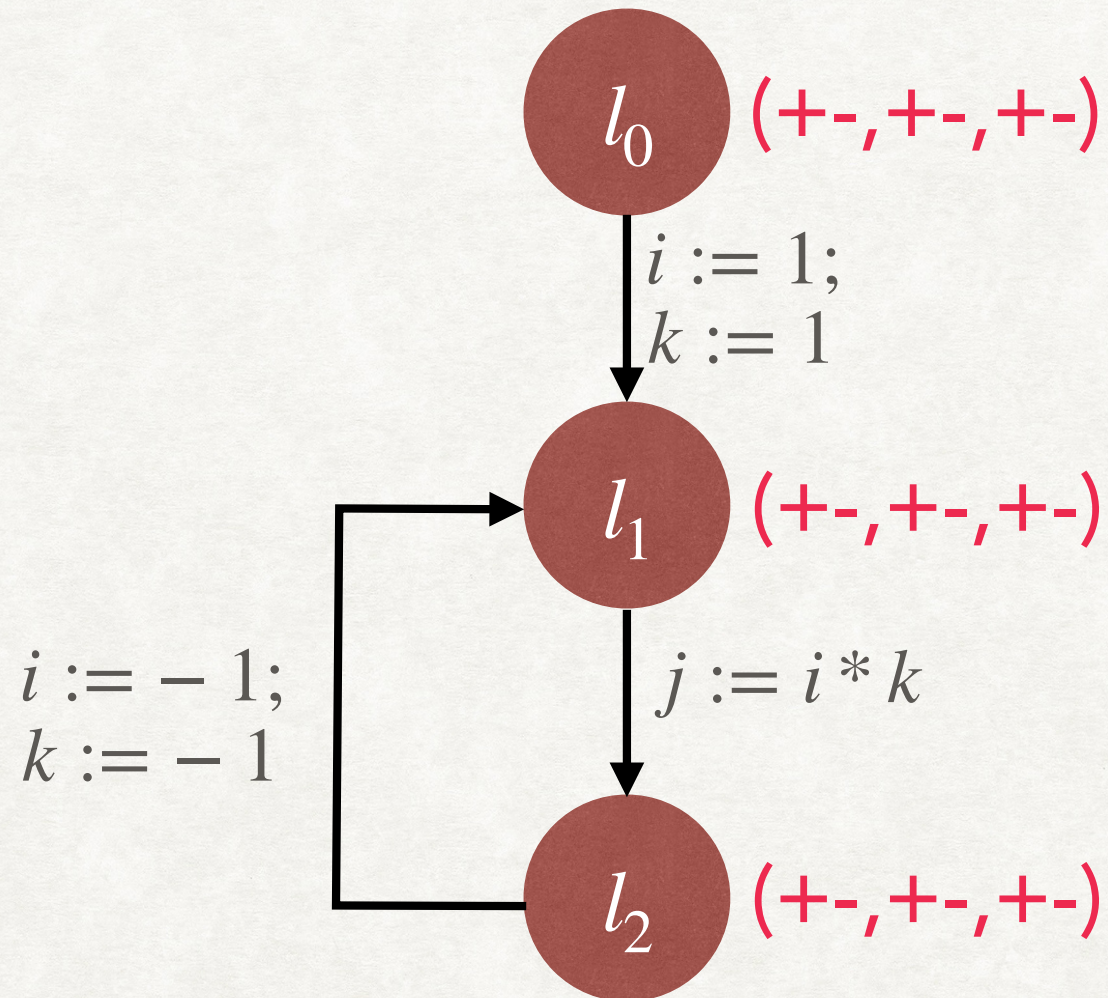


## EXAMPLE - KILDALL'S ALGORITHM



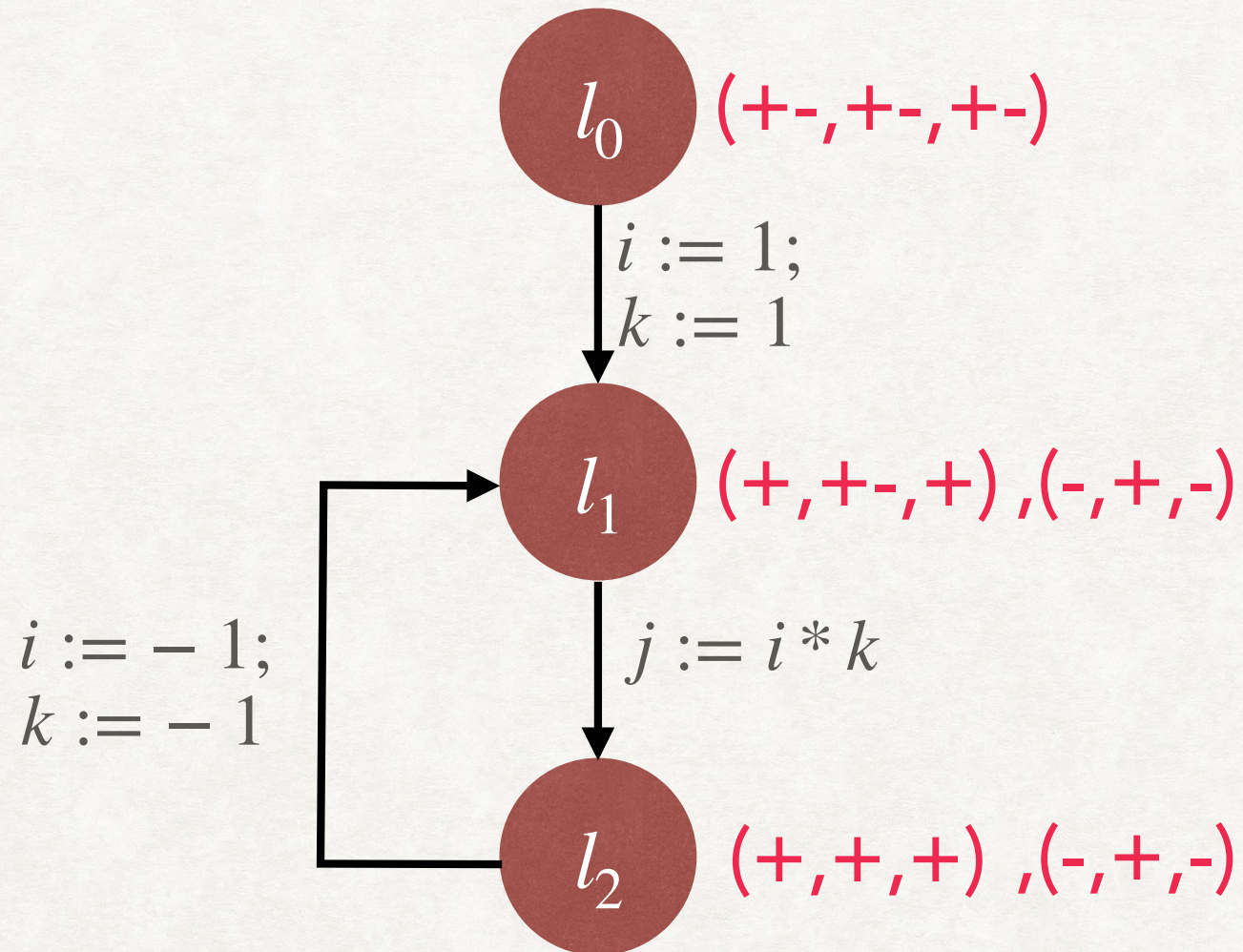


## EXAMPLE - KILDALL'S ALGORITHM



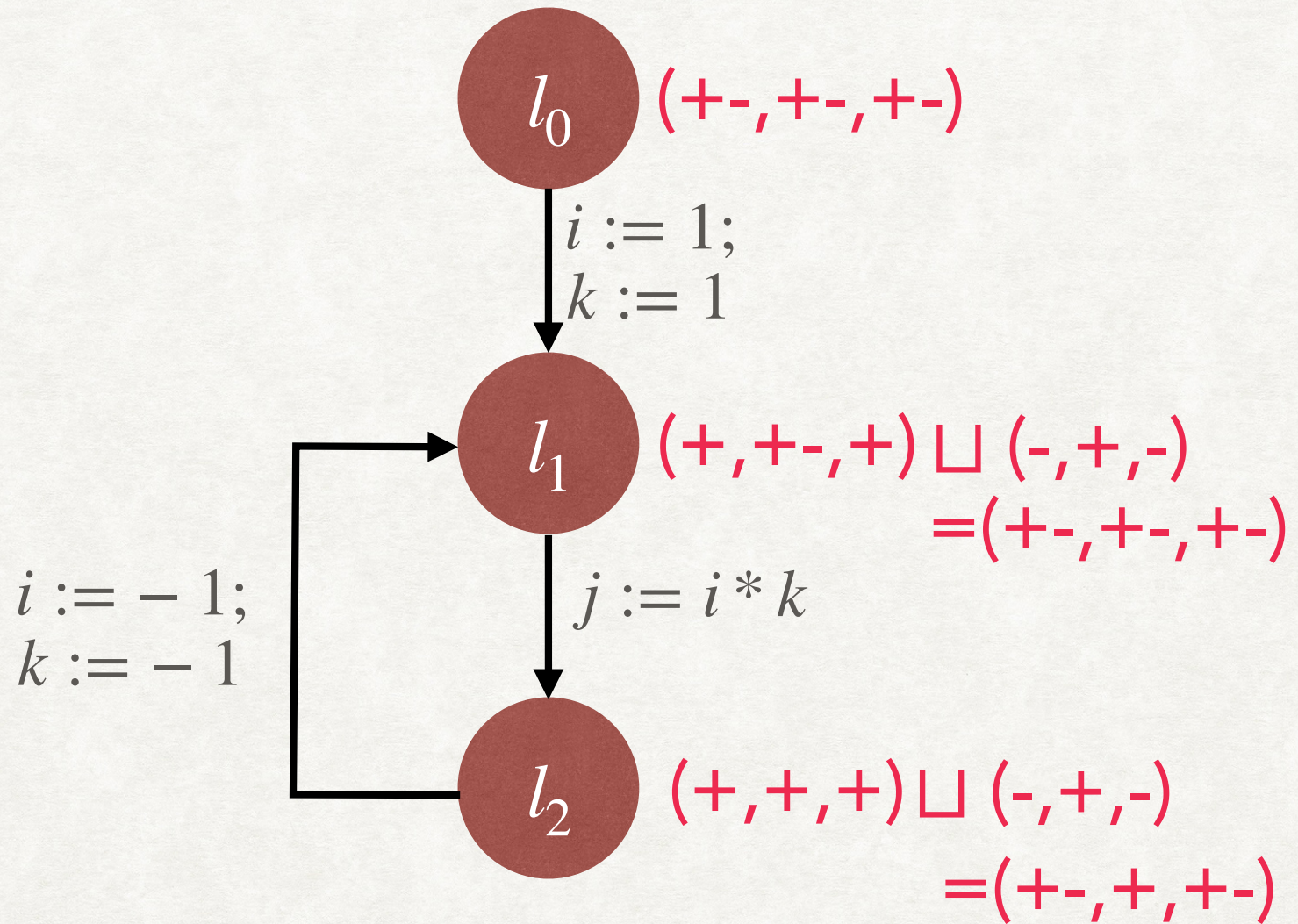


## EXAMPLE - ABSTRACT JOP



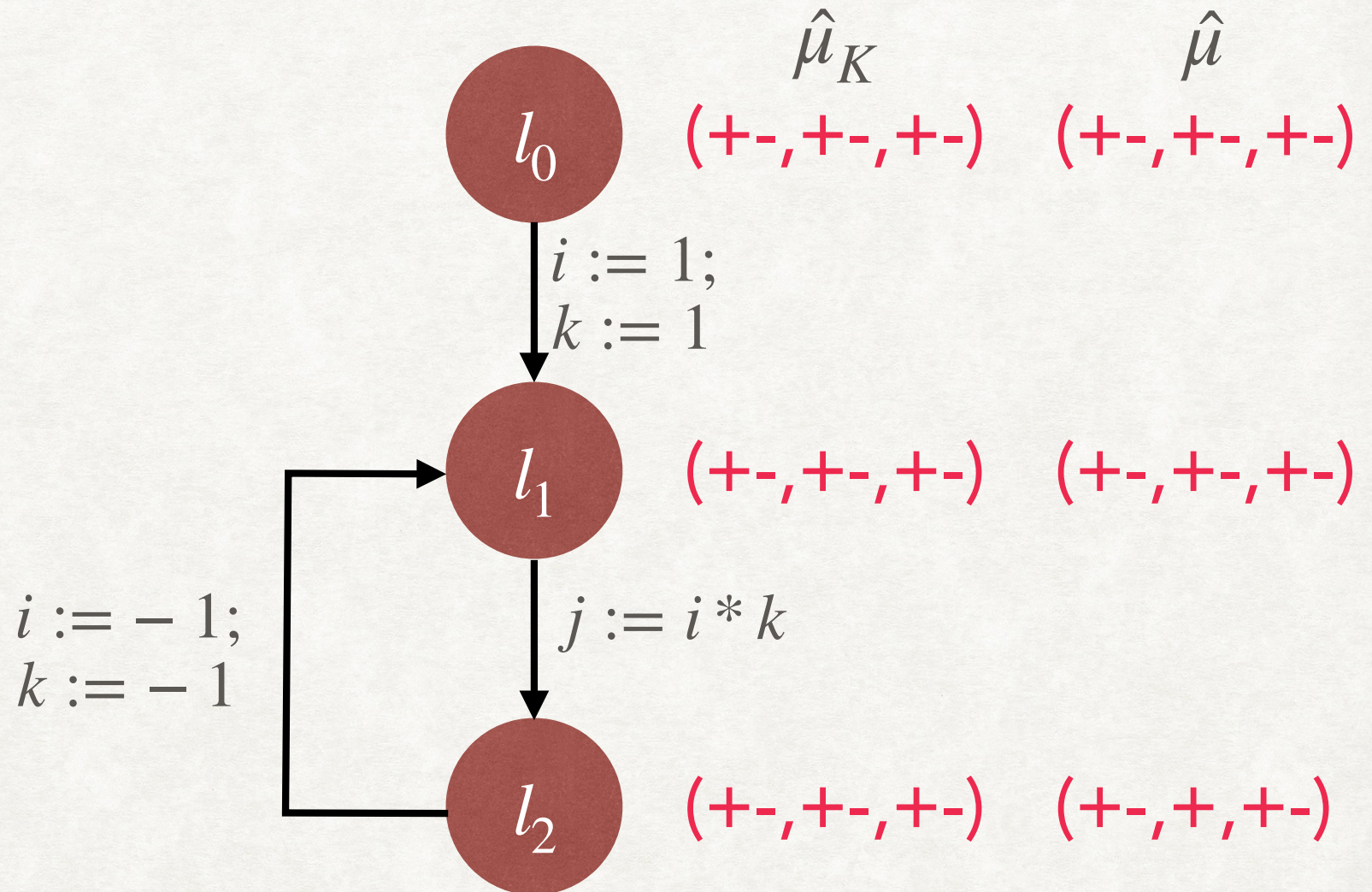


## EXAMPLE - ABSTRACT JOP





## EXAMPLE - KILDALL VS ABSTRACT JOP



$\hat{\mu}_K \neq \hat{\mu}$  : This is because Kildall's Algorithm applies join eagerly  
We will prove that  $\hat{\mu}_K \geq \hat{\mu}$



# PROPERTIES OF KILDALL'S ALGORITHM

1. The values computed using Kildall's algorithm are an over-approximation of the abstract JOP, if the underlying AI framework is monotonic.
2. In general, Kildall's algorithm computes the least solution to a system of equations.
3. If the abstract domain satisfies the ascending chain condition, then Kildall's algorithm is guaranteed to terminate.

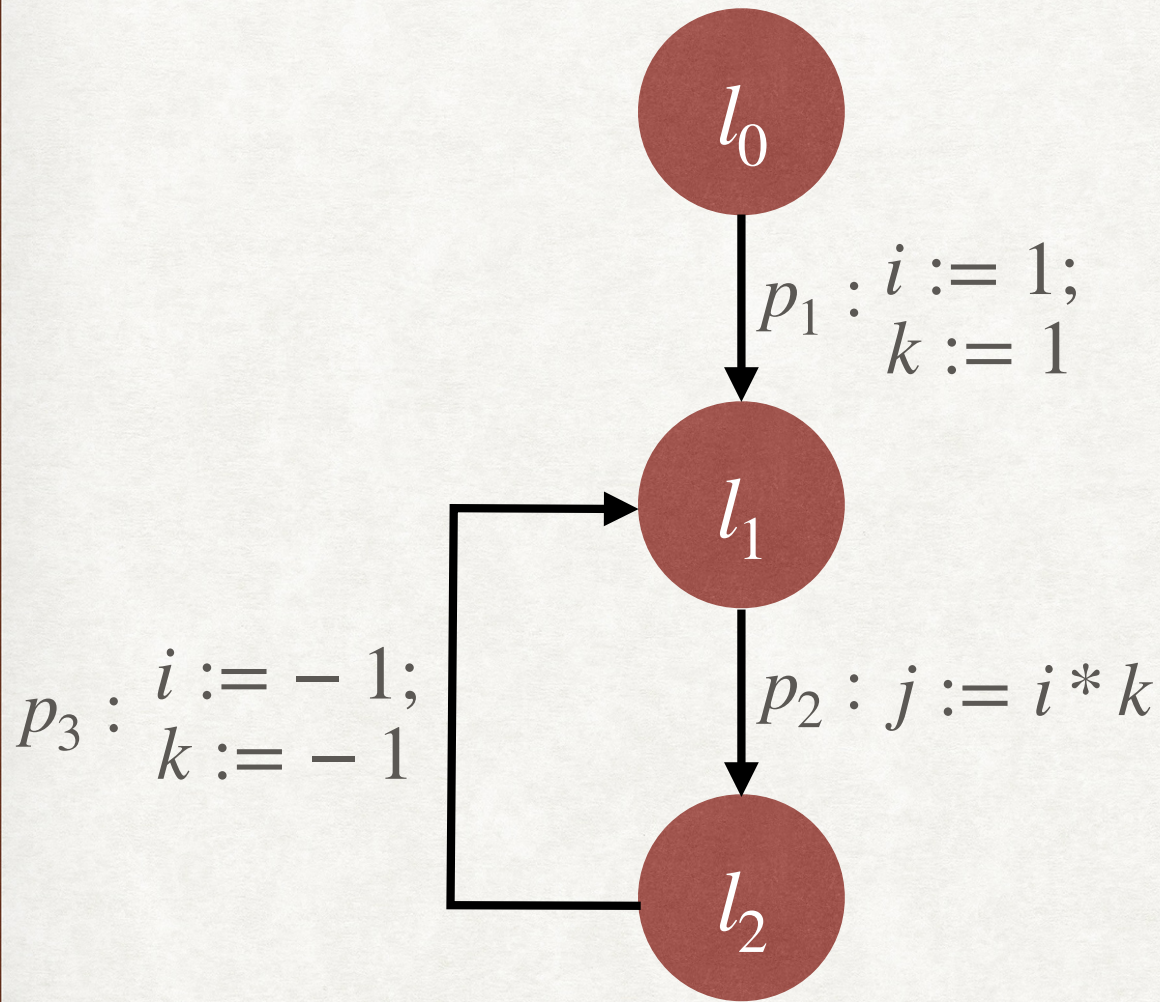


# DATAFLOW EQUATIONS

- Program  $\Gamma_c = (V, L, l_0, l_e, T)$  induces a system of data flow equations:
  - $X_{l_0} = d_0$
  - For all other locations  $l \in L \setminus \{l_0\}$ ,  $X_l = \bigsqcup_{(l', c, l) \in T} \hat{f}_c(X_{l'})$
- For collecting semantics, replace  $d_0$  with  $c_0$ ,  $\sqcup$  with  $\cup$  and  $\hat{f}_c$  with  $f_c$ .



## EXAMPLE - DATAFLOW EQUATIONS



$$X_{l_0} = d_0$$

$$X_{l_1} = \hat{f}_{p_1}(X_{l_0}) \sqcup \hat{f}_{p_3}(X_{l_2})$$

$$X_{l_2} = \hat{f}_{p_2}(X_{l_1})$$



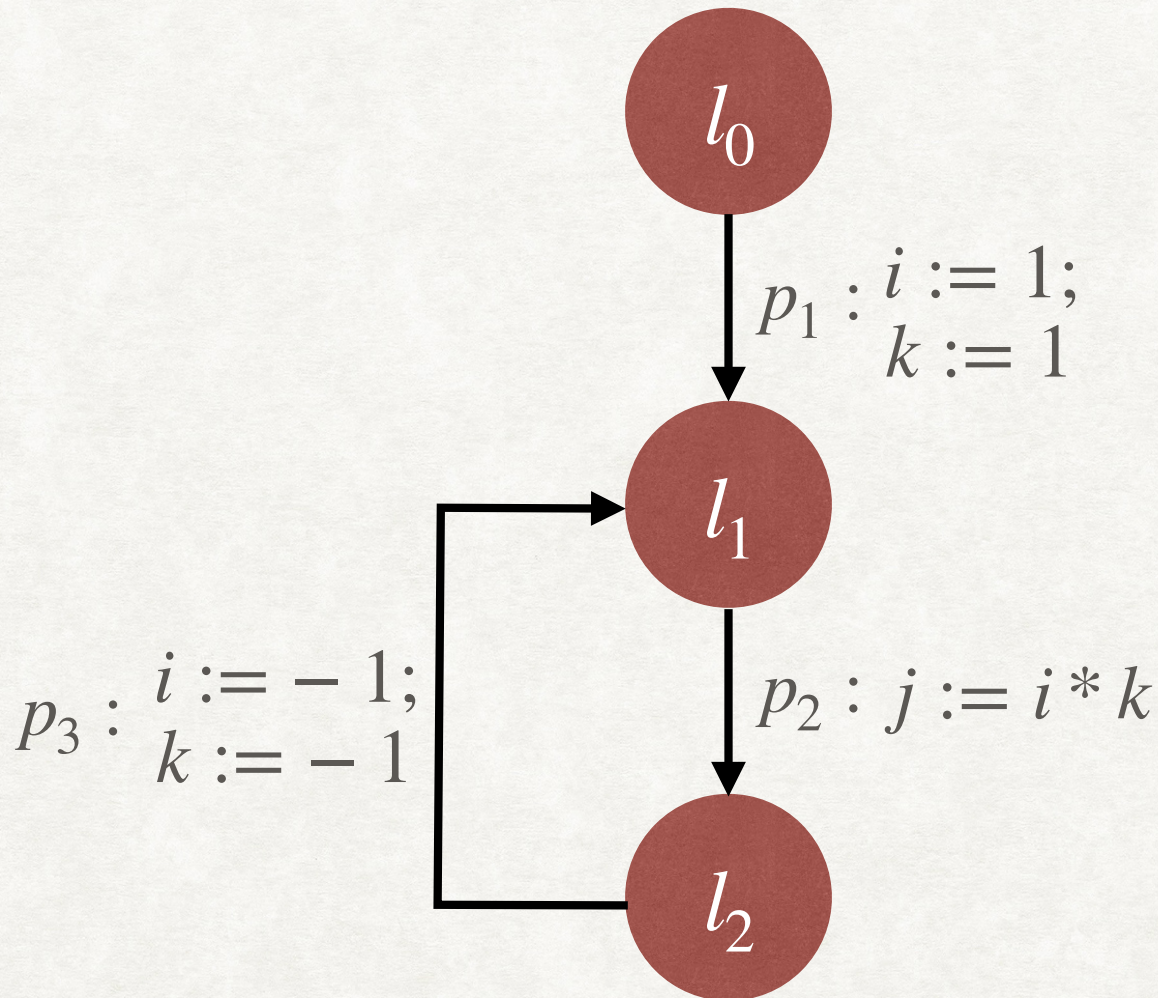
# DATAFLOW EQUATIONS AS FUNCTION

- Consider the 'vectorised' lattice  $(\bar{D}, \bar{\leq})$ , where  $\bar{D} = D^{|L|}$ .
  - $\bar{d} \bar{\leq} \bar{d}' \Leftrightarrow \forall l \in L. \bar{d}(l) \leq \bar{d}'(l)$
  - **Homework:** Prove that if  $(D, \leq)$  is a complete lattice, then  $(\bar{D}, \bar{\leq})$  is also a complete lattice.
- We can view the data flow equations as a function  $\bar{f}: \bar{D} \rightarrow \bar{D}$ :
  - $(\bar{f}(\bar{d}))(l_0) = d_0$
  - $(\bar{f}(\bar{d}))(l) = \bigsqcup_{(l', c, l) \in T} \hat{f}_c(\bar{d}(l'))$



# DATAFLOW EQUATIONS AS FUNCTION

## EXAMPLE



Notice that a  
fixpoint of  $\bar{f}$  is a  
solution to the  
dataflow equations

$$\bar{f}(d_{l_0}, d_{l_1}, d_{l_2}) = (d_0, \hat{f}_{p_1}(d_{l_0}) \sqcup \hat{f}_{p_3}(d_{l_2}), \hat{f}_{p_2}(d_{l_1}))$$



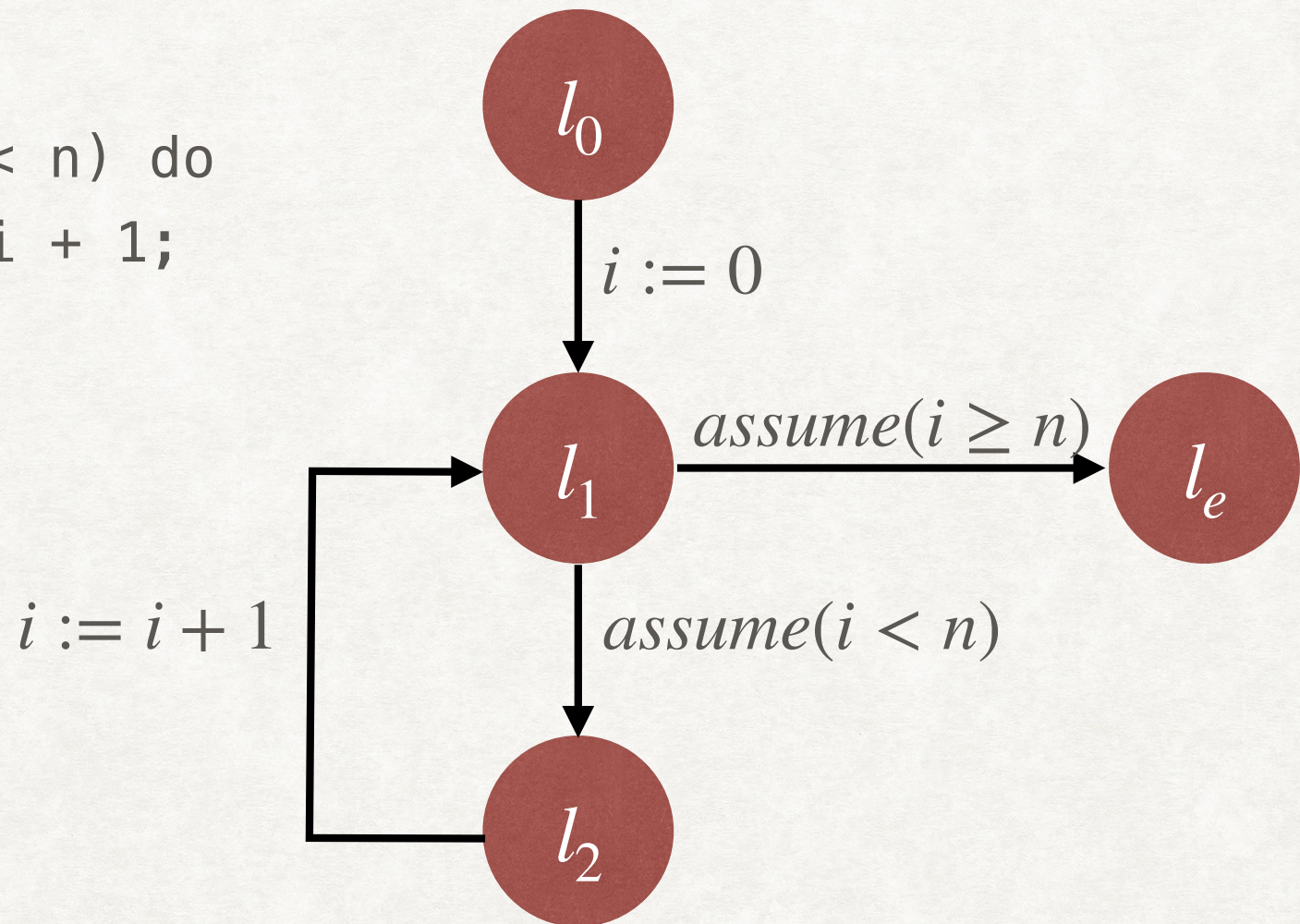
# DATAFLOW EQUATIONS AS FUNCTION

- If every abstract transfer function  $\hat{f} : D \rightarrow D$  is monotonic, then the function  $\bar{f} : \bar{D} \rightarrow \bar{D}$  is also monotonic.
  - **Homework:** Prove this.
- We have a monotonic function  $\bar{f}$  on a complete lattice  $\bar{D}$ . Hence, we can apply Knaster-Tarski fixpoint theorem.
- The least fixpoint  $lfp(\bar{f})$  exists, and is in fact the least solution to the dataflow equations.
- We will show that Kildall's algorithm actually computes  $lfp(\bar{f})$ .
- Note that we can also use the sequence  $\perp, \bar{f}(\perp), \bar{f}^2(\perp), \dots$  to compute  $lfp(\bar{f})$ .
  - This method is also called Kleene Iteration.



## LFP INTRODUCES THE LEAST OVER APPROXIMATION: EXAMPLE

```
i := 0;  
while(i < n) do  
  i := i + 1;
```



$(+ -, +, +, +)$  is a solution to the data flow equations,  
And  $(+ -, + -, + -, + -)$  is also another solution



# ABSTRACT JOP $\leq lfp(\bar{f})$ FOR MONOTONIC AI FRAMEWORK

## PROOF

- Given AI  $(D, \leq, \alpha, \gamma, \hat{F}_D)$ , if all functions in  $\hat{F}_D$  are monotonic, then Abstract JOP  $\leq lfp(\bar{f})$ .

**Proof:** Abstract JOP  $\hat{\mu}(l) = \bigsqcup_{\pi \in \Pi_l} \hat{f}_\pi(d_0)$

Let  $lfp(\bar{f}) = \bar{d}$ . We have to show that  $\forall l \in L. \hat{\mu}(l) \leq \bar{d}(l)$ .

We will show that for all locations  $l$ , all paths  $\pi \in \Pi_l$ ,  $\hat{f}_\pi(d_0) \leq \bar{d}(l)$ .

This proves the required result. Why?

We will use induction on length of the paths.

**Base Case:** Paths  $\pi$  of length 0 are empty and end at  $l_0$ . Hence,  $\hat{f}_\pi(d_0) = d_0$ .

Since  $\bar{f}(\bar{d}) = \bar{d}$  and  $(\bar{f}(\bar{d}))(l_0) = d_0$ , we have  $\bar{d}(l_0) = d_0$ .

Thus,  $\hat{f}_\pi(d_0) \leq \bar{d}(l_0)$

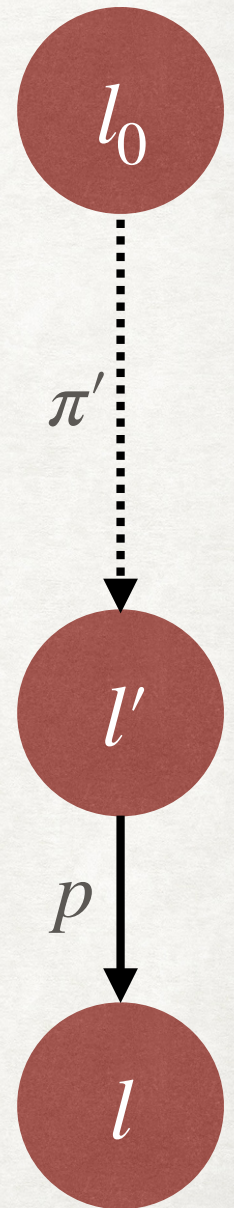


# ABSTRACT JOP $\leq lfp(\bar{f})$ FOR MONOTONIC AI FRAMEWORK

## PROOF

**Inductive Case:** Assume that the claim holds for all paths of length  $n$ .

Consider a path  $\pi$  of length  $n + 1$  ending at location  $l$ .





# ABSTRACT JOP $\leq lfp(\bar{f})$ FOR MONOTONIC AI FRAMEWORK

## PROOF

**Inductive Case:** Assume that the claim holds for all paths of length  $n$ .

Consider a path  $\pi$  of length  $n + 1$  ending at location  $l$ .

Let  $\pi'$  be the prefix of the path of length  $n$ , ending at location  $l'$ .

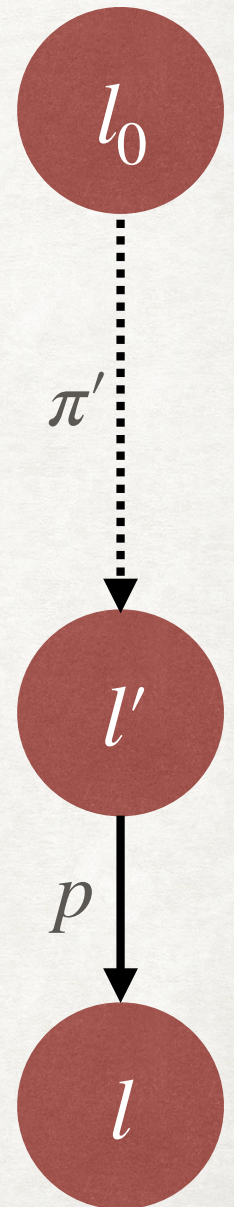
By Inductive Hypothesis,  $\hat{f}_{\pi'}(d_0) \leq \bar{d}(l')$ .

Since  $\hat{f}_p$  is monotonic,  $\hat{f}_p(\hat{f}_{\pi'}(d_0)) \leq \hat{f}_p(\bar{d}(l'))$ .

Hence,  $\hat{f}_{\pi}(d_0) \leq \hat{f}_p(\bar{d}(l'))$ .

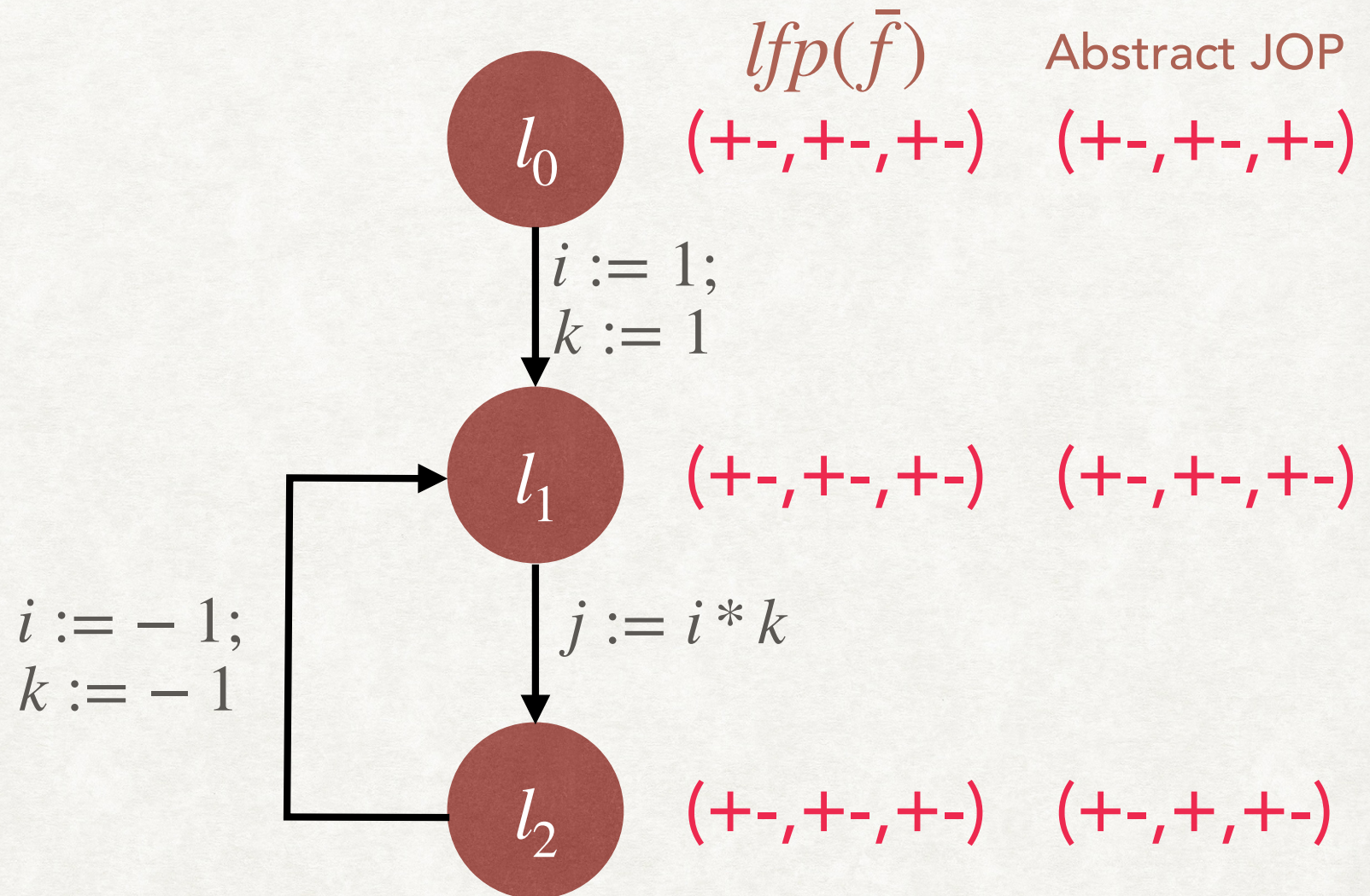
Now  $\bar{f}(\bar{d}) = \bar{d}$ . Hence,  $\bar{d}(l) = \bigsqcup_{(l',p,l) \in T} \hat{f}_p(\bar{d}(l'))$ .

Hence,  $\hat{f}_p(\bar{d}(l')) \leq \bar{d}(l)$ . Thus,  $\hat{f}_{\pi}(d_0) \leq \bar{d}(l)$ .





# EXAMPLE - LFP VS ABSTRACT JOP



$$\bar{f}(d_{l_0}, d_{l_1}, d_{l_2}) = (d_0, \hat{f}_{p_1}(d_{l_0}) \sqcup \hat{f}_{p_3}(d_{l_2}), \hat{f}_{p_2}(d_{l_1}))$$



# DISTRIBUTIVE AND INFINITELY DISTRIBUTIVE FUNCTIONS

- Given two posets  $(D_1, \leq_1)$  and  $(D_2, \leq_2)$ , function  $f: D_1 \rightarrow D_2$  is called distributive if for  $x, y \in D_1$  such that  $x \sqcup_1 y$  exists, then  $f(x) \sqcup_2 f(y)$  also exists, and  $f(x \sqcup_1 y) = f(x) \sqcup_2 f(y)$ .
- Given two posets  $(D_1, \leq_1)$  and  $(D_2, \leq_2)$ , function  $f: D_1 \rightarrow D_2$  is called infinitely distributive if for all  $X \subseteq D_1$  such that  $\sqcup_1 X$  exists, then  $\sqcup_2 f(X)$  also exists, and  $\sqcup_2 f(X) = f(\sqcup_1 X)$ .
- **Exercise:** If  $f$  is distributive, then  $f$  is also monotonic.



ABSTRACT JOP =  $lfp(\bar{f})$  FOR INFINITELY DISTRIBUTIVE AI FRAMEWORK

## PROOF

- Given AI  $(D, \leq, \alpha, \gamma, \hat{F}_D)$ , if all functions in  $\hat{F}_D$  are infinitely distributive, then Abstract JOP =  $lfp(\bar{f})$ .

**Proof:** We will show that Abstract JOP  $(\hat{\mu})$  is a fixpoint of  $\bar{f}$ . This is sufficient to prove the result. Why?



# ABSTRACT JOP = $lfp(\bar{f})$ FOR INFINITELY DISTRIBUTIVE AI FRAMEWORK

## PROOF

- Given AI  $(D, \leq, \alpha, \gamma, \hat{F}_D)$ , if all functions in  $\hat{F}_D$  are infinitely distributive, then Abstract JOP =  $lfp(\bar{f})$ .

**Proof:** We will show that Abstract JOP  $(\hat{\mu})$  is a fixpoint of  $\bar{f}$ . This is sufficient to prove the result. Why?

$$\begin{aligned}(\bar{f}(\hat{\mu}))(l) &= \bigsqcup_{(l',p,l) \in T} \hat{f}_p(\hat{\mu}(l')) \\ &= \bigsqcup_{(l',p,l) \in T} \hat{f}_p(\bigsqcup_{\pi \in \Pi_{l'}} \hat{f}_\pi(d_0)) \\ &= \bigsqcup_{(l',p,l) \in T} \bigsqcup_{\pi \in \Pi_{l'}} \hat{f}_p \circ \hat{f}_\pi(d_0)\end{aligned}$$



# ABSTRACT JOP = $lfp(\bar{f})$ FOR INFINITELY DISTRIBUTIVE AI FRAMEWORK

## PROOF

$$(\bar{f}(\hat{\mu}))(l) = \bigsqcup_{(l',p,l) \in T} \bigsqcup_{\pi \in \Pi_{l'}} \hat{f}_p \circ \hat{f}_\pi(d_0)$$

And we know that  $\hat{\mu}(l) = \bigsqcup_{\pi' \in \Pi_l} \hat{f}_{\pi'}(d_0)$ .

Then, due to associativity of  $\sqcup$ ,  $(\bar{f}(\hat{\mu}))(l) = \hat{\mu}(l)$ .

Thus,  $\hat{\mu}$  is a fixpoint of  $\bar{f}$ . We know from previous result that  $\hat{\mu} \leq lfp(\bar{f})$ . Thus,  $\hat{\mu} = lfp(\bar{f})$ .



## RECALL: SIGN ABSTRACT DOMAIN

- The abstract transfer functions in sign abstract domain are monotonic, but not infinitely distributive.

Consider  $p : j := i * k$  and  $d_1 = ( + , + - , + )$ ,  $d_2 = ( - , + - , - )$ .

Then,  $\hat{f}_p(d_1 \sqcup d_2) = ???$



## RECALL: SIGN ABSTRACT DOMAIN

- The abstract transfer functions in sign abstract domain are monotonic, but not infinitely distributive.

Consider  $p : j := i * k$  and  $d_1 = ( + , + - , + )$ ,  $d_2 = ( - , + - , - )$ .

Then,  $\hat{f}_p(d_1 \sqcup d_2) = ( + - , + - , + - )$ .



## RECALL: SIGN ABSTRACT DOMAIN

- The abstract transfer functions in sign abstract domain are monotonic, but not infinitely distributive.

Consider  $p : j := i * k$  and  $d_1 = ( + , + - , + )$ ,  $d_2 = ( - , + - , - )$ .

Then,  $\hat{f}_p(d_1 \sqcup d_2) = ( + - , + - , + - )$ .

$\hat{f}_p(d_1) \sqcup \hat{f}_p(d_2) = ???$



## RECALL: SIGN ABSTRACT DOMAIN

- The abstract transfer functions in sign abstract domain are monotonic, but not infinitely distributive.

Consider  $p : j := i * k$  and  $d_1 = (+, +-, +)$ ,  $d_2 = (-, +-, -)$ .

Then,  $\hat{f}_p(d_1 \sqcup d_2) = (+-, +-, +-)$ .

$$\hat{f}_p(d_1) \sqcup \hat{f}_p(d_2) = (+, +, +) \sqcup (-, +, -) = (+-, +, +-)$$



## RECALL: SIGN ABSTRACT DOMAIN

- The abstract transfer functions in sign abstract domain are monotonic, but not infinitely distributive.

Consider  $p : j := i * k$  and  $d_1 = (+, +-, +)$ ,  $d_2 = (-, +-, -)$ .

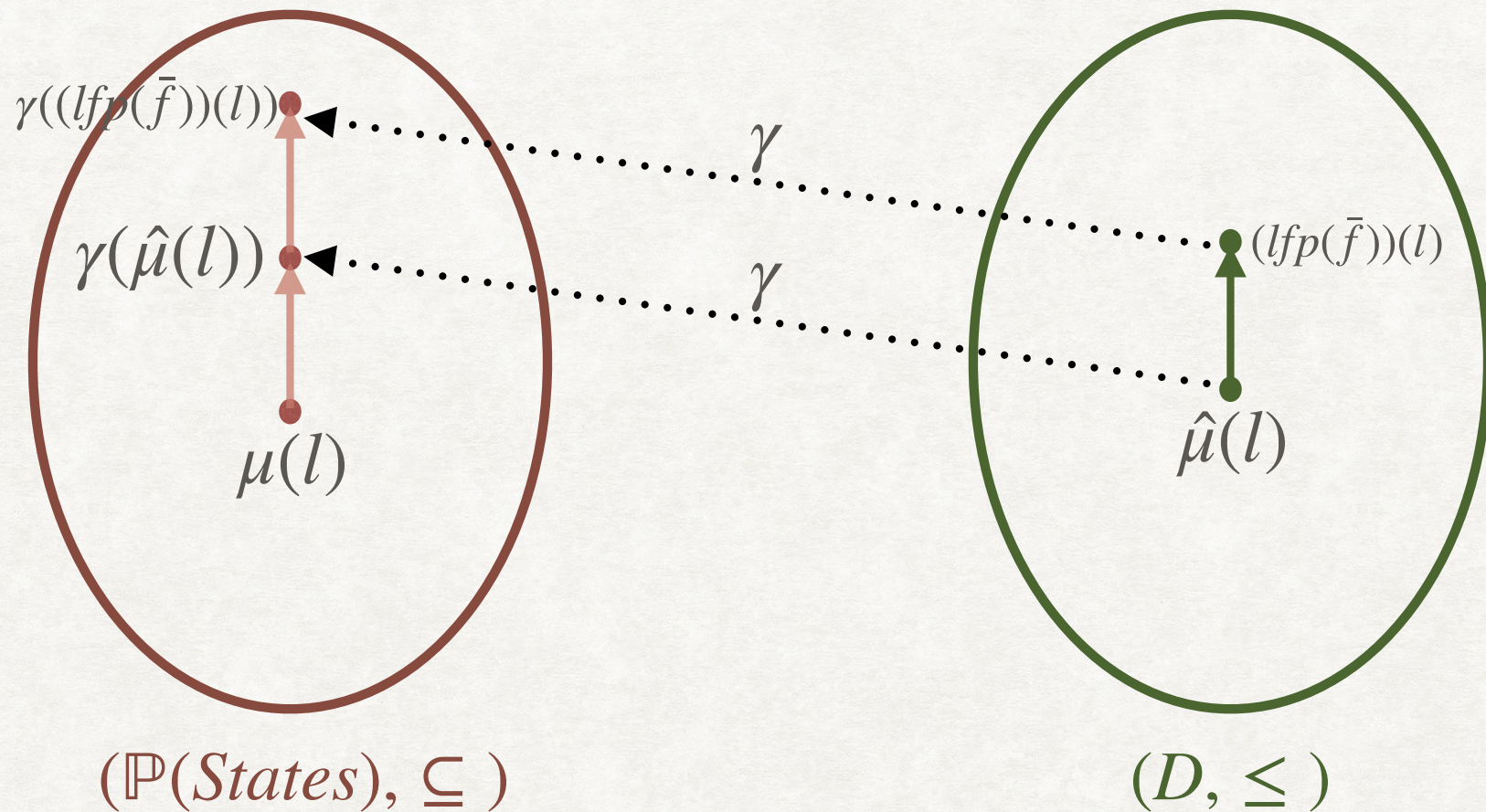
Then,  $\hat{f}_p(d_1 \sqcup d_2) = (+-, +-, +-)$ .

$$\hat{f}_p(d_1) \sqcup \hat{f}_p(d_2) = (+, +, +) \sqcup (-, +, -) = (+-, +, +-)$$

- The concrete transfer functions are always infinitely distributive. Hence, the concrete JOP is the least solution of the concrete data-flow equations.



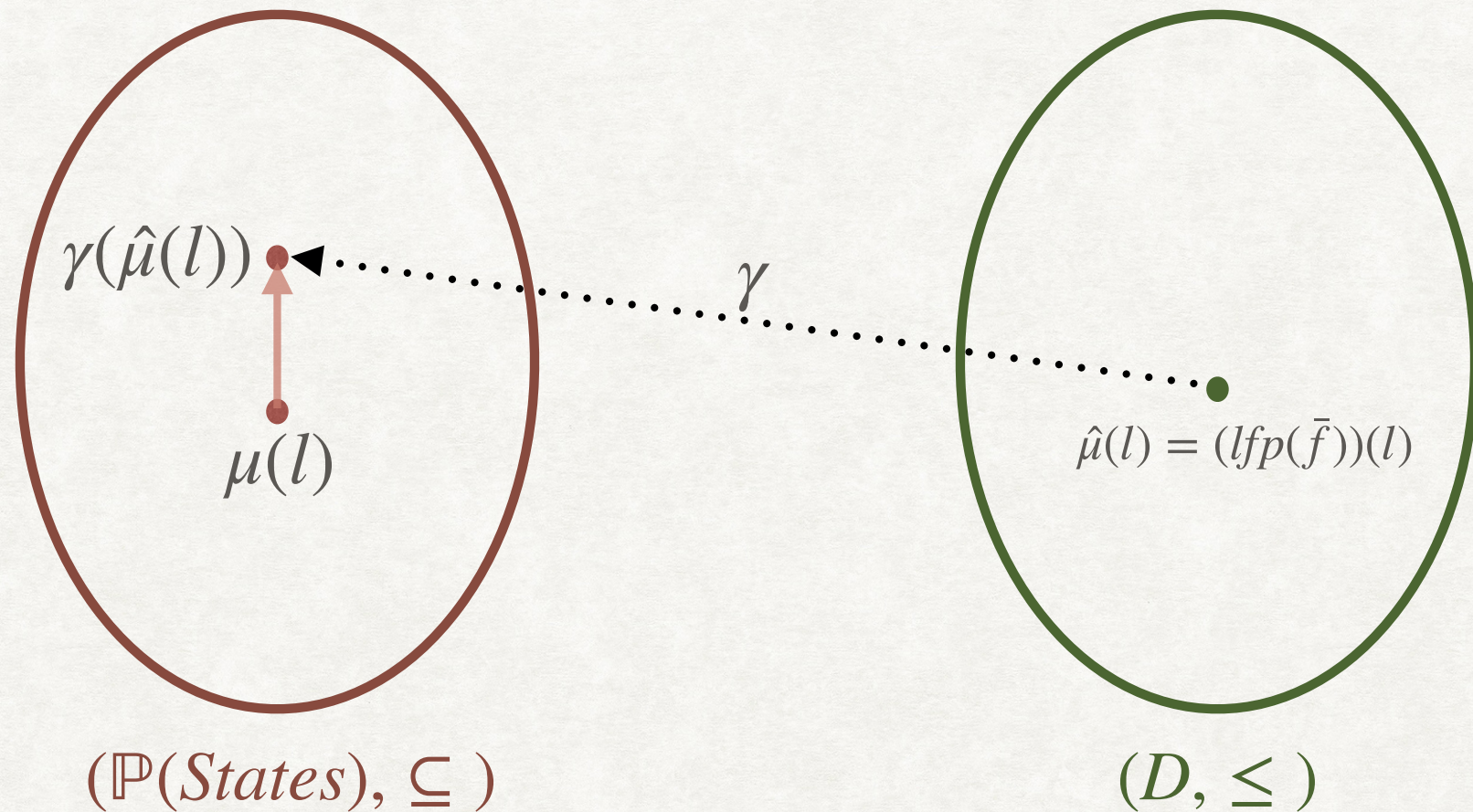
# BIG PICTURE



For Monotonic AI Framework



# BIG PICTURE



For Infinitely Distributive AI Framework



# KILDALL'S ALGORITHM COMPUTES $lfp(\bar{f})$

## PROOF

- First, we will show that  $\hat{\mu}_K \leq lfp(\bar{f})$

We will show that  $\hat{\mu}_K \leq lfp(\bar{f})$  is a loop invariant of the outer while loop.

At the beginning,  $\hat{\mu}_K(l_0) = \alpha(P) \leq d_0$ .

Hence,  $\forall l. \hat{\mu}_K(l) \leq (lfp(\bar{f}))(l)$ .

Assuming that the claim holds at the beginning of some iteration, let  $\hat{\mu}_K = \bar{d}$ ,  $lfp(\bar{f}) = \bar{g}$ . We have  $\bar{d} \leq \bar{g}$ .

AbstractForwardPropagate( $\Gamma_c, P$ )

$S := \{l_0\};$

$\hat{\mu}_K(l_0) := \alpha(P);$

$\hat{\mu}_K(l) := \perp, \text{ for } l \in L \setminus \{l_0\};$

while  $S \neq \emptyset$  do{

$l := \text{Choose } S;$

$S := S \setminus \{l\};$

    foreach  $(l, c, l') \in T$  do{

$F := \hat{f}_c(\hat{\mu}_K(l));$

        if  $\neg(F \leq \hat{\mu}_K(l'))$  then{

$\hat{\mu}_K(l') := \hat{\mu}_K(l') \sqcup F;$

$S := S \cup \{l'\};$

        }

    }

}



# KILDALL'S ALGORITHM COMPUTES $lfp(\bar{f})$

## PROOF

```
AbstractForwardPropagate( $\Gamma_c, P$ )  
   $S := \{l_0\};$   
   $\hat{\mu}_K(l_0) := \alpha(P);$   
   $\hat{\mu}_K(l) := \perp, \text{ for } l \in L \setminus \{l_0\};$   
  while  $S \neq \emptyset$  do{  
     $l := \text{Choose } S;$   
     $S := S \setminus \{l\};$   
    foreach  $(l, c, l') \in T$  do{  
       $F := \hat{f}_c(\hat{\mu}_K(l));$   
      if  $\neg(F \leq \hat{\mu}_K(l'))$  then{  
         $\hat{\mu}_K(l') := \hat{\mu}_K(l') \sqcup F;$   
         $S := S \cup \{l'\};$   
      }  
    }  
  }
```



# KILDALL'S ALGORITHM COMPUTES $lfp(\bar{f})$

## PROOF

For some successor  $l'$  of  $l$ ,  
 $\hat{\mu}_K(l') = d(l') \sqcup \hat{f}_c(d(l))$ .

Now,  $\bar{d}(l) \leq \bar{g}(l) \Rightarrow \hat{f}_c(\bar{d}(l)) \leq \hat{f}_c(\bar{g}(l))$ .

Further,  $\bar{g}(l') = \bigsqcup_{(l,c,l') \in T} \hat{f}_c(\bar{g}(l))$

Hence,  $\bar{g}(l') \geq \hat{f}_c(\bar{g}(l)) \geq \hat{f}_c(\bar{d}(l))$

We also know that  $\bar{g}(l') \geq \bar{d}(l')$ .

Thus,  $\bar{g}(l') \geq \bar{d}(l') \sqcup \hat{f}_c(\bar{d}(l))$ .

Hence,  $\bar{g}(l') \geq \hat{\mu}_K(l')$ .

AbstractForwardPropagate( $\Gamma_c, P$ )

$S := \{l_0\};$

$\hat{\mu}_K(l_0) := \alpha(P);$

$\hat{\mu}_K(l) := \perp$ , for  $l \in L \setminus \{l_0\};$

while  $S \neq \emptyset$  do{

$l := \text{Choose } S;$

$S := S \setminus \{l\};$

    foreach  $(l, c, l') \in T$  do{

$F := \hat{f}_c(\hat{\mu}_K(l));$

        if  $\neg(F \leq \hat{\mu}_K(l'))$  then{

$\hat{\mu}_K(l') := \hat{\mu}_K(l') \sqcup F;$

$S := S \cup \{l'\};$

        }

    }

}



# KILDALL'S ALGORITHM COMPUTES $lfp(\bar{f})$

## PROOF

Next, we will show that  $\hat{\mu}_K \geq lfp(\bar{f})$ .

To prove this, we will show that when the algorithm terminates, the final  $\hat{\mu}_K$  is a post-fixpoint of  $\bar{f}$ , i.e.  $\bar{f}(\hat{\mu}_K) \leq \hat{\mu}_K$ .

Then, by Knaster-Tarski theorem,  $lfp(\bar{f})$  is the glb of all post-fixpoints, and hence the claim follows.

We will prove that following is a loop invariant of the outer while-loop:

$$\begin{aligned} \forall l \in L \setminus S. \forall l' \in L. (l, c, l') \in T \\ \Rightarrow \hat{\mu}_K(l') \geq \hat{f}_c(\hat{\mu}_K(l)) \end{aligned}$$

AbstractForwardPropagate( $\Gamma_c, P$ )

```
S := {l0};  
 $\hat{\mu}_K(l_0) := \alpha(P)$ ;  
 $\hat{\mu}_K(l) := \perp$ , for  $l \in L \setminus \{l_0\}$ ;  
while S  $\neq \emptyset$  do{  
    l := Choose S;  
    S := S  $\setminus$  {l};  
    foreach (l, c, l')  $\in$  T do{  
        F :=  $\hat{f}_c(\hat{\mu}_K(l))$ ;  
        if  $\neg(F \leq \hat{\mu}_K(l'))$  then{  
             $\hat{\mu}_K(l') := \hat{\mu}_K(l') \sqcup F$ ;  
            S := S  $\cup$  {l'};  
        }  
    }  
}
```



# KILDALL'S ALGORITHM COMPUTES $lfp(\bar{f})$

## PROOF

$\forall l \in L \setminus S. \forall l' \in L. (l, c, l') \in T$

$\Rightarrow \hat{\mu}_K(l') \geq \hat{f}_c(\hat{\mu}_K(l))$

AbstractForwardPropagate( $\Gamma_c, P$ )

$S := \{l_0\};$

$\hat{\mu}_K(l_0) := \alpha(P);$

$\hat{\mu}_K(l) := \perp, \text{ for } l \in L \setminus \{l_0\};$

while  $S \neq \emptyset$  do{

$l := \text{Choose } S;$

$S := S \setminus \{l\};$

    foreach  $(l, c, l') \in T$  do{

$F := \hat{f}_c(\hat{\mu}_K(l));$

        if  $\neg(F \leq \hat{\mu}_K(l'))$  then{

$\hat{\mu}_K(l') := \hat{\mu}_K(l') \sqcup F;$

$S := S \cup \{l'\};$

        }

    }

}



# KILDALL'S ALGORITHM COMPUTES $lfp(\bar{f})$

## PROOF

$$\forall l \in L \setminus S. \forall l' \in L. (l, c, l') \in T$$

$$\Rightarrow \hat{\mu}_K(l') \geq \hat{f}_c(\hat{\mu}_K(l))$$

On exiting the loop, we will have

$$\forall l, l' \in L. (l, c, l') \in T \Rightarrow \hat{\mu}_K(l') \geq \hat{f}_c(\hat{\mu}_K(l))$$

$$\Rightarrow \forall l' \in L. \hat{\mu}_K(l') \geq \bigsqcup_{(l, c, l') \in T} \hat{f}_c(\hat{\mu}_K(l))$$

$$\Rightarrow \forall l' \in L. \hat{\mu}_K(l') \geq (\bar{f}(\hat{\mu}_K))(l')$$

AbstractForwardPropagate( $\Gamma_c, P$ )

$S := \{l_0\};$

$\hat{\mu}_K(l_0) := \alpha(P);$

$\hat{\mu}_K(l) := \perp, \text{ for } l \in L \setminus \{l_0\};$

while  $S \neq \emptyset$  do{

$l := \text{Choose } S;$

$S := S \setminus \{l\};$

    foreach  $(l, c, l') \in T$  do{

$F := \hat{f}_c(\hat{\mu}_K(l));$

        if  $\neg(F \leq \hat{\mu}_K(l'))$  then{

$\hat{\mu}_K(l') := \hat{\mu}_K(l') \sqcup F;$

$S := S \cup \{l'\};$

        }

    }

}



# KILDALL'S ALGORITHM COMPUTES $lfp(\bar{f})$

## PROOF

$$\forall l \in L \setminus S. \forall l' \in L. (l, c, l') \in T$$

$$\Rightarrow \hat{\mu}_K(l') \geq \hat{f}_c(\hat{\mu}_K(l))$$

At the beginning, the invariant holds, assuming that  $\hat{f}_c(\perp) = \perp$ .

Note that if  $\hat{f}_c(\perp) \neq \perp$ , we can initialise  $S$  with  $L$ .

AbstractForwardPropagate( $\Gamma_c, P$ )

```

S := {l0};
μK(l0) := α(P);
μK(l) := ⊥, for l ∈ L \ {l0};
while S ≠ ∅ do{
    l := Choose S;
    S := S \ {l};
    foreach (l, c, l') ∈ T do{
        F := f̂c(μK(l));
        if ¬(F ≤ μK(l')) then{
            μK(l') := μK(l') ⊔ F;
            S := S ∪ {l'};
        }
    }
}
    
```



# KILDALL'S ALGORITHM COMPUTES $lfp(\bar{f})$

## PROOF

$\forall l \in L \setminus S. \forall l' \in L. (l, c, l') \in T$

$$\Rightarrow \hat{\mu}_K(l') \geq \hat{f}_c(\hat{\mu}_K(l))$$

Assume that the claim holds at the beginning of some iteration.

For each successor  $l'$  of  $l$ , either  $\hat{\mu}_K(l') \geq \hat{f}_c(\hat{\mu}_K(l))$ , or we enter the if-body and re-assign  $\hat{\mu}_K(l')$  to ensure that  $\hat{\mu}_K(l') \geq \hat{f}_c(\hat{\mu}_K(l))$ .

Thus, the loop invariant continues to hold.

This concludes the proof that the final  $\hat{\mu}_K = lfp(\bar{f})$ .

AbstractForwardPropagate( $\Gamma_c, P$ )

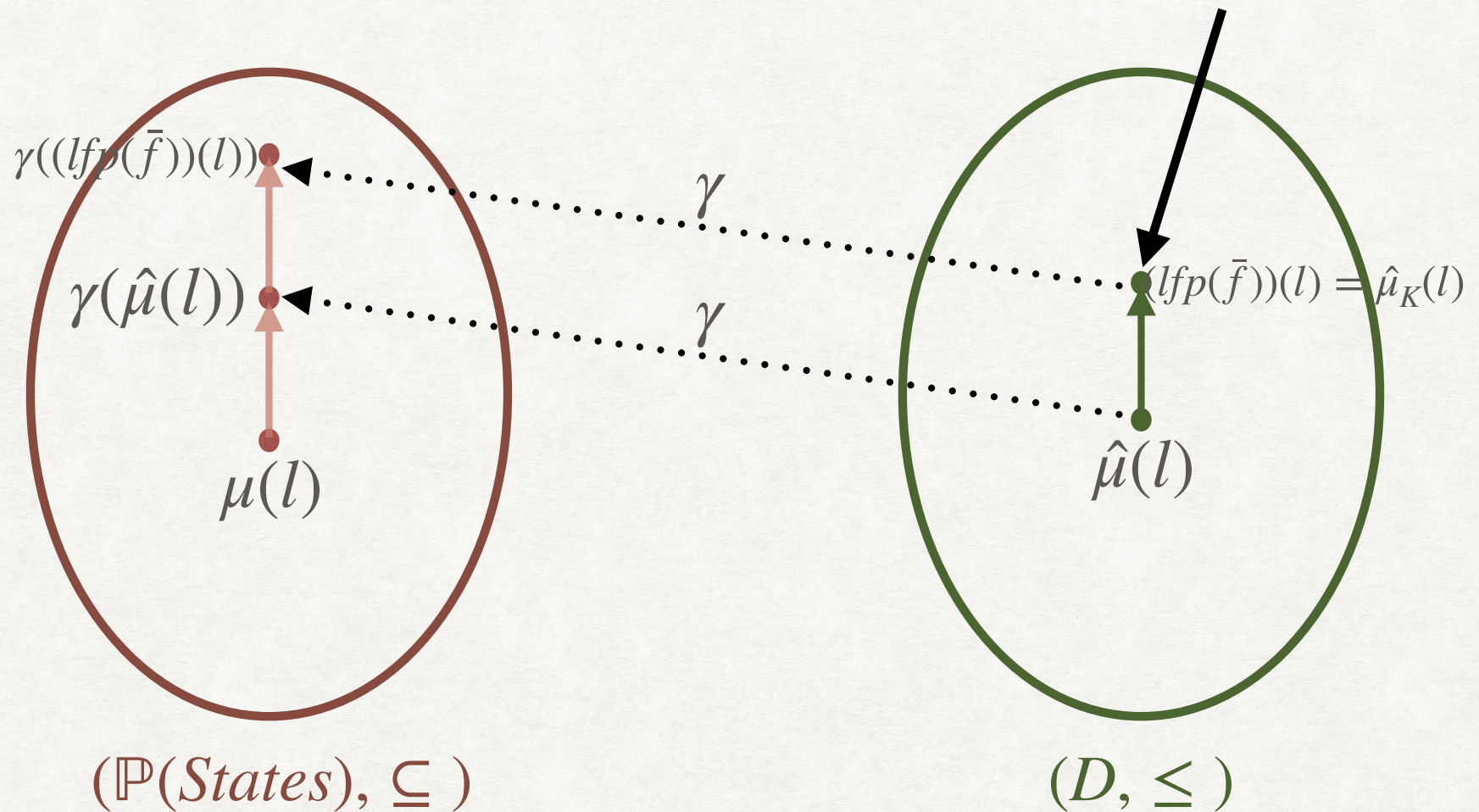
```

S := {l0};
μK(l0) := α(P);
μK(l) := ⊥, for l ∈ L \ {l0};
while S ≠ ∅ do{
    l := Choose S;
    S := S \ {l};
    foreach (l, c, l') ∈ T do{
        F := fc(μK(l));
        if ¬(F ≤ μK(l')) then{
            μK(l') := μK(l') ⊔ F;
            S := S ∪ {l'};
        }
    }
}
    
```



# BIG PICTURE

Kildall's Algorithm computes this



For Monotonic AI Framework



# KILDALL'S ALGORITHM: TERMINATION

- Consider the vector of values maintained by the algorithm across locations.
- After each iteration of the outer loop, either this vector increases or it stays the same and  $S$  decreases.
- If  $(D, \leq)$  satisfies the ascending chain condition, then so does  $(\bar{D}, \bar{\leq})$ .
  - In this case, the loop is guaranteed to terminate.

```

AbstractForwardPropagate( $\Gamma_c, P$ )
   $S := \{l_0\};$ 
   $\hat{\mu}_K(l_0) := \alpha(P);$ 
   $\hat{\mu}_K(l) := \perp, \text{ for } l \in L \setminus \{l_0\};$ 
  while  $S \neq \emptyset$  do{
     $l := \text{Choose } S;$ 
     $S := S \setminus \{l\};$ 
    foreach  $(l, c, l') \in T$  do{
       $F := \hat{f}_c(\hat{\mu}_K(l));$ 
      if  $\neg(F \leq \hat{\mu}_K(l'))$  then{
         $\hat{\mu}_K(l') := \hat{\mu}_K(l') \sqcup F;$ 
         $S := S \cup \{l'\};$ 
      }
    }
  }
  }
    
```



# KILDALL'S ALGORITHM

## SUFFICIENT CONDITIONS

- Kildall's Algorithm can be used with an abstract domain  $(D, \leq, \alpha, \gamma, \hat{F}_D)$  if:
  - $(D, \leq)$  is a complete lattice.
  - $(\mathbb{P}(\text{State}), \subseteq) \xrightleftharpoons[\gamma]{\alpha} (D, \leq)$
  - Every abstract transfer function in  $\hat{F}_D$  is a consistent abstraction of the corresponding concrete transfer function.
  - Every abstract transfer function in  $\hat{F}_D$  is monotonic.
  - $(D, \leq)$  satisfies the ascending chain condition.



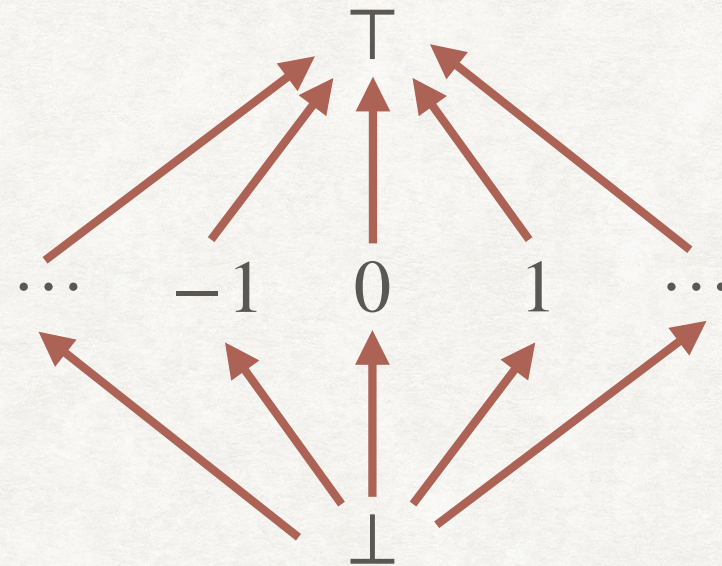
## APPLYING KILDALL'S ALGORITHM USING CONCRETE PROGRAM STATES

- Recall the concrete lattice of program states:  $(\mathbb{P}(\text{States}), \subseteq)$  where  $\text{States} = \text{Var} \rightarrow \mathbb{Z}$ .
- Does this lattice satisfy ACC?
- Kildall's Algorithm using concrete lattice  $\equiv$  ForwardPropagate Algorithm.
  - Since the concrete lattice does not satisfy ACC, termination of Kildall's Algorithm is not guaranteed.
- Since the concrete transfer functions are infinitely distributive, LFP = JOP.



# CONSTANT ABSTRACT DOMAIN

- $I = \mathbb{Z} \cup \{ \top, \perp \}$ 
  - $\forall n \in \mathbb{Z}. \perp \leq n \leq \top$
  - Flat, but infinite lattice.
  - Satisfies ACC.
- $D = V \rightarrow I$





# CONSTANT ABSTRACT DOMAIN

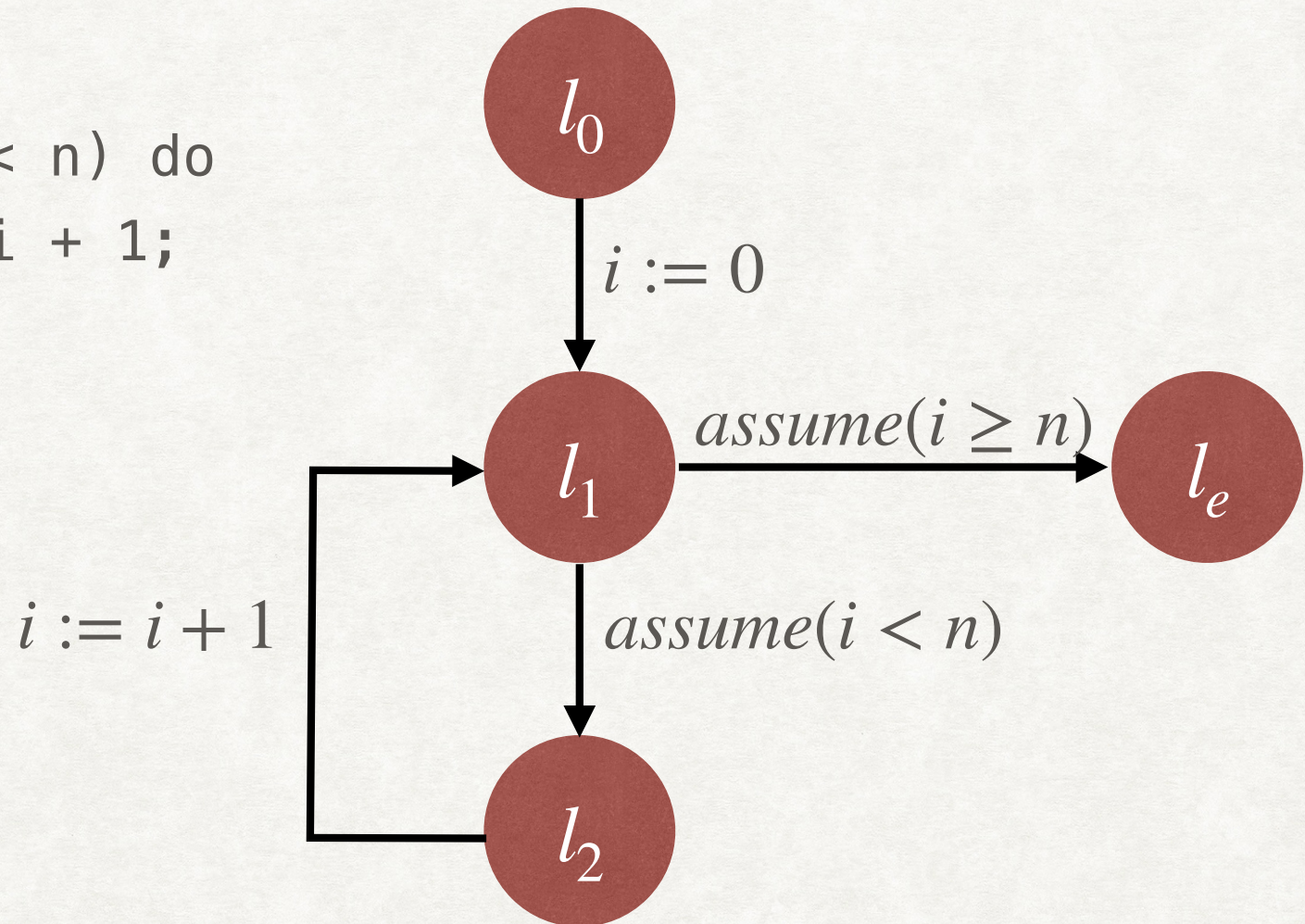
## ABSTRACTION AND CONCRETIZATION FUNCTION

- $\alpha(c) = d$ 
  - If  $c = \emptyset$ , then  $\forall v. d(v) = \perp$
  - Otherwise,  $d(v) = \begin{cases} n & \text{if } \forall \alpha \in c. \sigma(v) = n \\ \top & \text{otherwise} \end{cases}$
- $\gamma(d) = \{\sigma \mid \forall v \in V. \forall n \in \mathbb{Z}. d(v) = n \rightarrow \sigma(v) = n\}$
- $\alpha$  and  $\gamma$  form an onto Galois connection.



# COMPUTING ABSTRACT JOP VERSUS LFP: EXAMPLE

```
i := 0;  
while(i < n) do  
  i := i + 1;
```



Algorithm for computing the abstract JOP will never terminate  
However, due to ACC, LFP computation is guaranteed to terminate



# CONSTANT ABSTRACT DOMAIN

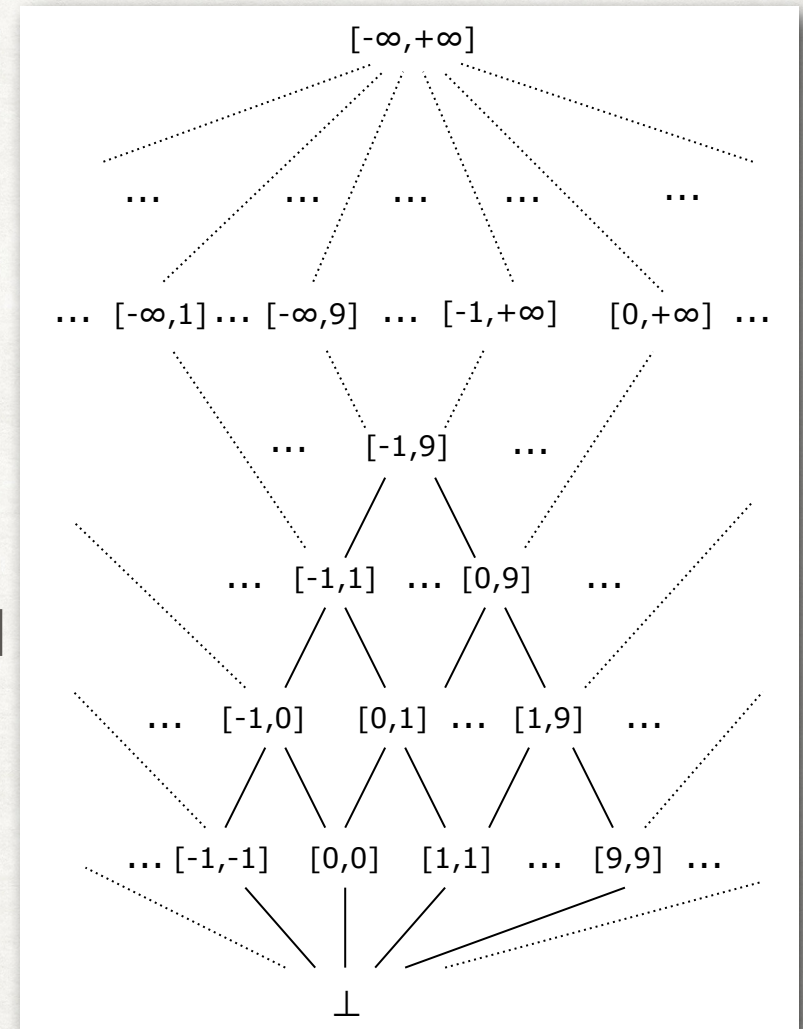
## ABSTRACT TRANSFER FUNCTIONS

- What will be  $\hat{f}_c$  for  $c : x := e$ ?
  - Is  $\hat{f}_c$  distributive?
- What will be  $\hat{f}_c$  for  $c : \text{assume}(x = n)$ ?



# INTERVAL ABSTRACT DOMAIN

- $I = \{[a, b] \mid a, b \in \mathbb{Z} \cup \{-\infty, \infty\}\} \cup \{\perp\}$ 
  - $D = V \rightarrow I$
  - Also called Box abstract domain.
- $[a_1, b_1] \sqsubseteq [a_2, b_2] \Leftrightarrow a_2 \leq a_1 \wedge b_1 \leq b_2,$   
 $\forall d \in I. \perp \sqsubseteq d$
- Is  $(I, \sqsubseteq)$  a lattice?
  - $[a_1, b_1] \sqcup [a_2, b_2] = [\min(a_1, a_2), \max(b_1, b_2)]$
- Is  $(I, \sqsubseteq)$  a complete lattice?
  - Maximal element?
- $(D, \sqsubseteq)$ :  
 $\forall d_1, d_2 \in D. d_1 \sqsubseteq d_2 \Leftrightarrow \forall v \in V. d_1(v) \sqsubseteq d_2(v)$





# INTERVAL ABSTRACT DOMAIN

## ABSTRACTION AND CONCRETIZATION FUNCTION

- $\alpha : \mathbb{P}(\text{States}) \rightarrow D, \gamma : D \rightarrow \mathbb{P}(\text{States})$
- $\alpha(c) = d$ 
  - $d(v) = [\min\{\sigma(v) \mid \sigma \in c\}, \max\{\sigma(v) \mid \sigma \in c\}]$
- $\gamma(d) = \{\sigma \mid \forall v \in V. d(v) = [a, b] \Rightarrow a \leq \sigma(v) \leq b\}$
- Is  $(\mathbb{P}(\text{States}), \subseteq) \xrightleftharpoons[\gamma]{\alpha} (D, \sqsubseteq)$  a Galois Connection?
  - Is it an Onto Galois Connection?



# INTERVAL ABSTRACT DOMAIN

## ABSTRACT TRANSFER FUNCTION

- Consider  $c : x := x + y$ 
  - We can use interval arithmetic for  $\hat{f}_c$
- Assuming  $d(x) = [l_x, u_x], d(y) = [l_y, u_y]$ 
  - $\hat{f}_c(d) = d[x \mapsto [l_x + l_y, u_x + u_y]]$
- Is  $\hat{f}_c$  distributive?
  - What about  $\hat{f}_c$  for  $c : x := x - y$ ? Is this function distributive?