



# Static Program Analysis

**Lecture 7: Dataflow Analysis VI  
(Undecidability of MOP Solution & Non-ACC Domains)**

**Winter Semester 2016/17**

**Thomas Noll  
Software Modeling and Verification Group  
RWTH Aachen University**

<https://moves.rwth-aachen.de/teaching/ws-1617/spa/>

## Bringen Sie Informatik zur Wirkung!

Serviceorientiert, zustandslos, geschichtet, lose gekoppelt, ad hoc entstanden oder model-driven... was ist die beste Softwarearchitektur?

### Inhalte des Workshops:

- Anhand eines realen Beispiels aus der Praxis entwickelt ihr in kleinen Teams die Architektur für ein Informationssystem.
- Da sich Software fortlaufend weiterentwickelt, prüfen wir, wie sich die Architektur gegenüber neuen, auch unerwarteten Anforderungen verhält und daran anpassen lässt. Gemeinsam diskutieren wir den Einfluss verschiedener Anforderungen, mögliche Varianten und ihre Auswirkungen.
- Außerdem zeigen wir euch die Aufgaben eines Softwarearchitekten und wie er Projekte zum Erfolg führt.



Mittwoch, **14.12.2016**, 09.00-16.00 Uhr

Seminarraum 5053.2, B-IT Research School, RWTH Aachen (Ahornstraße 55)

Melden Sie sich unter Angabe Ihres Semesters bis zum **4.12.2016** an.  
Wir freuen uns auf Sie!

Kontakt: Anne-Kristin Hauk ([hauk@itestra.de](mailto:hauk@itestra.de)) – [www.itestra.com](http://www.itestra.com)

# Recap: The MOP Solution

---

## The MOP Solution

### Definition (MOP solution)

Let  $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$  be a dataflow system where  $Lab = \{l_1, \dots, l_n\}$ .  
The **MOP solution** for  $S$  is determined by

$$\text{mop}(S) := (\text{mop}(l_1), \dots, \text{mop}(l_n)) \in D^n$$

where, for every  $l \in Lab$ ,

$$\text{mop}(l) := \bigsqcup \{ \varphi_\pi(\iota) \mid \pi \in Path(l) \}.$$

### Remark:

- $Path(l)$  is generally infinite
- ⇒ not clear how to compute  $\text{mop}(l)$
- In fact: MOP solution generally undecidable (later)

# Recap: The MOP Solution

## MOP vs. Fixpoint Solution I

### Example (Constant Propagation)

```
c := if [z > 0]1 then
    [x := 2]2; [y := 3]3
else
    [x := 3]4; [y := 2]5
end;
[z := x+y]6; [...]7
```

#### Transfer functions

(for  $\delta = (\delta(x), \delta(y), \delta(z)) \in D$ ):

$$\varphi_1(a, b, c) = (a, b, c)$$

$$\varphi_2(a, b, c) = (2, b, c)$$

$$\varphi_3(a, b, c) = (a, 3, c)$$

$$\varphi_4(a, b, c) = (3, b, c)$$

$$\varphi_5(a, b, c) = (a, 2, c)$$

$$\varphi_6(a, b, c) = (a, b, a + b)$$

#### 1. Fixpoint solution:

$$CP_1 = \iota = (\top, \top, \top)$$

$$CP_2 = \varphi_1(CP_1) = (\top, \top, \top)$$

$$CP_3 = \varphi_2(CP_2) = (2, \top, \top)$$

$$CP_4 = \varphi_1(CP_3) = (\top, \top, \top)$$

$$CP_5 = \varphi_4(CP_4) = (3, \top, \top)$$

$$CP_6 = \varphi_3(CP_3) \sqcup \varphi_5(CP_5) \\ = (2, 3, \top) \sqcup (3, 2, \top) = (\top, \top, \top)$$

$$CP_7 = \varphi_6(CP_6) = (\top, \top, \top)$$

#### 2. MOP solution:

$$\text{mop}(7) = \varphi_{[1,2,3,6]}(\top, \top, \top) \sqcup \\ \varphi_{[1,4,5,6]}(\top, \top, \top) \\ = (2, 3, 5) \sqcup (3, 2, 5) \\ = (\top, \top, 5)$$

## Recap: The MOP Solution

### MOP vs. Fixpoint Solution II

#### Theorem (MOP vs. Fixpoint Solution)

Let  $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$  be a dataflow system. Then

$$\text{mop}(S) \sqsubseteq \text{fix}(\Phi_S)$$

**Reminder:** by Definition 4.9,

$$\Phi_S : D^n \rightarrow D^n : (d_1, \dots, d_n) \mapsto (d'_1, \dots, d'_n)$$

where  $Lab = \{1, \dots, n\}$  and, for each  $l \in Lab$ ,

$$d'_l := \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup \{ \varphi_{l'}(d_{l'}) \mid (l', l) \in F \} & \text{otherwise} \end{cases}$$

Proof.

on the board □

**Remark:** as Example ?? shows,  $\text{mop}(S) \neq \text{fix}(\Phi_S)$  is possible

## Recap: The MOP Solution

---

### Distributivity of Transfer Functions

A sufficient condition for the coincidence of MOP and Fixpoint Solution is the distributivity of the transfer functions.

#### Definition (Distributivity)

- Let  $(D, \sqsubseteq)$  and  $(D', \sqsubseteq')$  be complete lattices. Function  $F : D \rightarrow D'$  is called **distributive** (w.r.t.  $(D, \sqsubseteq)$  and  $(D', \sqsubseteq')$ ) if, for every  $d_1, d_2 \in D$ ,

$$F(d_1 \sqcup_D d_2) = F(d_1) \sqcup_{D'} F(d_2).$$

- A dataflow system  $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$  is called **distributive** if every  $\varphi_l : D \rightarrow D$  ( $l \in Lab$ ) is so.

# Recap: The MOP Solution

---

## Coincidence of MOP and Fixpoint Solution

### Theorem (MOP vs. Fixpoint Solution)

Let  $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$  be a distributive dataflow system. Then

$$\text{mop}(S) = \text{fix}(\Phi_S)$$

### Proof.

- $\text{mop}(S) \sqsubseteq \text{fix}(\Phi_S)$ : Theorem ??
- $\text{fix}(\Phi_S) \sqsubseteq \text{mop}(S)$ : as  $\text{fix}(\Phi_S)$  is the *least* fixpoint of  $\Phi_S$ , it suffices to show that  $\Phi_S(\text{mop}(S)) = \text{mop}(S)$  (on the board)



# Undecidability of the MOP Solution

---

## Undecidability of the MOP Solution I

### Theorem 7.1 (Undecidability of MOP solution)

*The MOP solution for Constant Propagation is undecidable.*

### Proof.

Based on undecidability of **Modified Post Correspondence Problem**:

Let  $\Gamma$  be some alphabet,  $n \in \mathbb{N}$ , and  $u_1, \dots, u_n, v_1, \dots, v_n \in \Gamma^+$ .

Do there exist  $i_1, \dots, i_m \in \{1, \dots, n\}$  with  $m \geq 1$  and  $i_1 = 1$  such that

$$u_{i_1} u_{i_2} \dots u_{i_m} = v_{i_1} v_{i_2} \dots v_{i_m}?$$

Given a MPCP, we construct a WHILE program (with strings and Booleans) whose MOP analysis detects a constant property iff the MPCP has no solution (see next slide). □



# Undecidability of the MOP Solution

## Undecidability of the MOP Solution II

Proof (continued).

```
x := u1; y := v1;
while ... do
  if ... then
    x := x ++ u1;
    y := y ++ v1
  else if ... then
    ⋮
  else
    x := x ++ un;
    y := y ++ vn
  end ... end
end;
z := (x = y);
[skip]'
```

Then:  $\text{mop}(l)(z) = \text{false}$

$\iff x \neq y$  at the end of every path to  $l$

$\iff$  the MPCP has no solution

□

# Dataflow Analysis with Non-ACC Domains

---

## Dataflow Analysis with Non-ACC Domains

- **Reminder:**  $(D, \sqsubseteq)$  satisfies **ACC** if each ascending chain  $d_0 \sqsubseteq d_1 \sqsubseteq \dots$  eventually stabilises, i.e., there exists  $n \in \mathbb{N}$  such that  $d_n = d_{n+1} = \dots$
- If **height** (= maximal chain size - 1) of  $(D, \sqsubseteq)$  is  $m$ , then fixpoint computation terminates after at most  $|Lab| \cdot m$  iterations
- **But:** if  $(D, \sqsubseteq)$  has **non-stabilising ascending chains**  
     $\implies$  algorithm may not terminate
- **Solution:** use **widening operators** to enforce termination

## Example: Interval Analysis

### Example: Interval Analysis

#### Interval Analysis

The goal of **Interval Analysis** is to determine, for each (interesting) program point, a safe interval for the values of the (interesting) program variables.

Interval analysis is actually a generalisation of constant propagation ( $\approx$  interval analysis with 1-element intervals)

#### Example 7.2 (Interval Analysis)

```
var a[100]: int;
i := 0;
while i <= 42 do
  if i >= 0 ^ i < 100 then
    a[i] := i
  end;
  i := i + 1;
end;
```

← redundant array bounds check

## Example: Interval Analysis

---

### The Domain of Interval Analysis

- The domain  $(Int, \subseteq)$  of **intervals over**  $\mathbb{Z}$  is defined by

$$Int := \{[z_1, z_2] \mid z_1 \in \mathbb{Z} \cup \{-\infty\}, z_2 \in \mathbb{Z} \cup \{+\infty\}, z_1 \leq z_2\} \cup \{\emptyset\}$$

where

- $-\infty \leq z$  and  $z \leq +\infty$  (for all  $z \in \mathbb{Z}$ )
- $\emptyset \subseteq J$  (for all  $J \in Int$ )
- $[y_1, y_2] \subseteq [z_1, z_2]$  iff  $y_1 \geq z_1$  and  $y_2 \leq z_2$
- $(Int, \subseteq)$  is a **complete lattice** with (for every  $\mathcal{I} \subseteq Int$ )

$$\bigsqcup \mathcal{I} = \begin{cases} \emptyset & \text{if } \mathcal{I} = \emptyset \text{ or } \mathcal{I} = \{\emptyset\} \\ [z_1, z_2] & \text{otherwise} \end{cases}$$

where

$$z_1 := \bigcap_{\mathbb{Z} \cup \{-\infty\}} \{z_1 \mid [z_1, z_2] \in \mathcal{I}\}$$
$$z_2 := \bigcup_{\mathbb{Z} \cup \{+\infty\}} \{z_2 \mid [z_1, z_2] \in \mathcal{I}\}$$

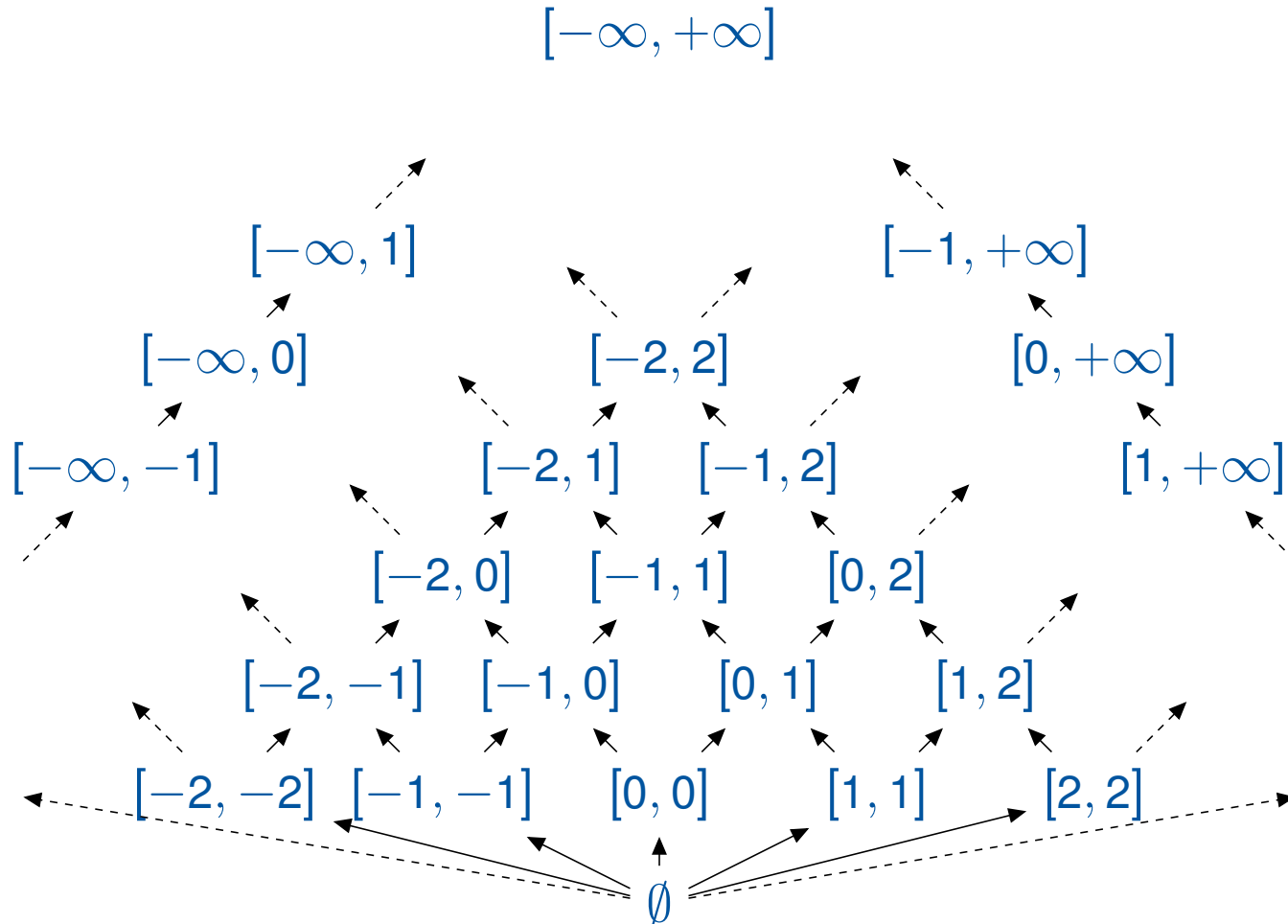
(and thus  $\perp = \emptyset$ ,  $\top = [-\infty, +\infty]$ )

- Clearly  $(Int, \subseteq)$  has **infinite ascending chains**, such as

$$\emptyset \subseteq [1, 1] \subseteq [1, 2] \subseteq [1, 3] \subseteq \dots$$

# Example: Interval Analysis

## The Complete Lattice of Interval Analysis



# Formalising Interval Analysis

---

## Formalising Interval Analysis I

The **dataflow system**  $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$  is given by

- set of labels  $Lab := Lab_c$
- extremal labels  $E := \{init(c)\}$  (forward problem)
- flow relation  $F := flow(c)$  (forward problem)
- complete lattice  $(D, \sqsubseteq)$  where
  - $D := \{\delta \mid \delta : Var_c \rightarrow Int\}$
  - $\delta_1 \sqsubseteq \delta_2$  iff  $\delta_1(x) \subseteq \delta_2(x)$  for every  $x \in Var_c$
- $\iota := \top_D : Var_c \rightarrow Int : x \mapsto \top_{Int}$  (with  $\top_{Int} = [-\infty, +\infty]$ )
- $\varphi$ : see next slide

# Formalising Interval Analysis

---

## Formalising Interval Analysis II

Transfer functions  $\{\varphi_l \mid l \in Lab\}$  are defined by

$$\varphi_l(\delta) := \begin{cases} \delta & \text{if } B' = \text{skip or } B' \in BExp \\ \delta[x \mapsto val_\delta(a)] & \text{if } B' = (x := a) \end{cases}$$

where

$$\begin{array}{ll} val_\delta(x) := \delta(x) & val_\delta(a_1 + a_2) := val_\delta(a_1) \oplus val_\delta(a_2) \\ val_\delta(z) := [z, z] & val_\delta(a_1 - a_2) := val_\delta(a_1) \ominus val_\delta(a_2) \\ & val_\delta(a_1 * a_2) := val_\delta(a_1) \odot val_\delta(a_2) \end{array}$$

with

$$\begin{aligned} \emptyset \oplus J &:= J \oplus \emptyset := \emptyset \ominus J := \dots := \emptyset \\ [y_1, y_2] \oplus [z_1, z_2] &:= [y_1 + z_1, y_2 + z_2] \\ [y_1, y_2] \ominus [z_1, z_2] &:= [y_1 - z_2, y_2 - z_1] \\ [y_1, y_2] \odot [z_1, z_2] &:= [\bigsqcap \{y_1 z_1, y_1 z_2, y_2 z_1, y_2 z_2\}, \bigsqcup \{y_1 z_1, y_1 z_2, y_2 z_1, y_2 z_2\}] \end{aligned}$$

# Formalising Interval Analysis

---

## Remarks

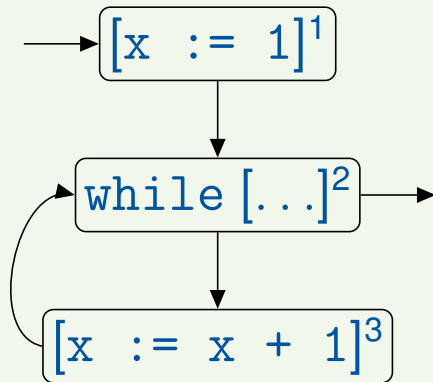
- Possible **refinement of DFA** to take conditional blocks  $b'$  ( $b \in BExp$ ) into account
  - essentially:  $b$  as edge label,  $\varphi_1(\delta)(x) = \delta(x) \setminus \{z \in \mathbb{Z} \mid x = z \implies \neg b\}$   
(cf. “DFA with Conditional Branches” later)
- Important: **soundness and optimality** of abstract operations, e.g.,  $\oplus$ :
  - soundness:  $z_1 \in J_1, z_2 \in J_2 \implies z_1 + z_2 \in J_1 \oplus J_2$
  - optimality:  $J_1 \oplus J_2$  as precise (i.e., small) as possible



# Formalising Interval Analysis

## Interval Analysis without Widening

### Example 7.3



Transfer functions (for  $\delta(\mathbf{x}) = \mathcal{J}$ ):

$$\varphi_1(\mathcal{J}) = [1, 1]$$

$$\varphi_2(\mathcal{J}) = \mathcal{J}$$

$$\varphi_3(\emptyset) = \emptyset$$

$$\varphi_3([x_1, x_2]) = [x_1 + 1, x_2 + 1]$$

Application of worklist algorithm without widening: does not terminate (on the board)

# Applying Widening to Interval Analysis

## Widening Operators

### Definition 7.4 (Widening operator)

Let  $(D, \sqsubseteq)$  be a complete lattice. A mapping  $\nabla : D \times D \rightarrow D$  is called **widening operator** if

- for every  $d_1, d_2 \in D$ ,

$$d_1 \sqcup d_2 \sqsubseteq d_1 \nabla d_2$$

and

- for all ascending chains  $d_0 \sqsubseteq d_1 \sqsubseteq \dots$ , the ascending chain  $d_0^\nabla \sqsubseteq d_1^\nabla \sqsubseteq \dots$  eventually stabilises where

$$d_0^\nabla := d_0 \text{ and } d_{i+1}^\nabla := d_i^\nabla \nabla d_{i+1} \text{ for each } i \in \mathbb{N}$$

### Remarks:

- $(d_i^\nabla)_{i \in \mathbb{N}}$  is clearly an **ascending chain** as  $d_{i+1}^\nabla = d_i^\nabla \nabla d_{i+1} \sqsupseteq d_i^\nabla \sqcup d_{i+1} \sqsupseteq d_i^\nabla$
- In contrast to  $\sqcup$ ,  $\nabla$  does *not* have to be commutative, associative, monotonic, nor absorptive ( $d \nabla d = d$ )
- The requirement  $d_1 \sqcup d_2 \sqsubseteq d_1 \nabla d_2$  guarantees **soundness** of widening

# Applying Widening to Interval Analysis

## Applying Widening to Interval Analysis

- A **widening operator**:  $\nabla : Int \times Int \rightarrow Int$  with

$$\begin{aligned}\emptyset \nabla J &:= J \nabla \emptyset := J \\ [x_1, x_2] \nabla [y_1, y_2] &:= [z_1, z_2] \quad \text{where} \\ z_1 &:= \begin{cases} x_1 & \text{if } x_1 \leq y_1 \\ -\infty & \text{otherwise} \end{cases} \\ z_2 &:= \begin{cases} x_2 & \text{if } x_2 \geq y_2 \\ +\infty & \text{otherwise} \end{cases}\end{aligned}$$

- Widening turns infinite ascending chain

$$J_0 = \emptyset \subseteq J_1 = [1, 1] \subseteq J_2 = [1, 2] \subseteq J_3 = [1, 3] \subseteq \dots$$

into a finite one:

$$\begin{aligned}J_0^\nabla &= J_0 = \emptyset \\ J_1^\nabla &= J_0^\nabla \nabla J_1 = \emptyset \nabla [1, 1] = [1, 1] \\ J_2^\nabla &= J_1^\nabla \nabla J_2 = [1, 1] \nabla [1, 2] = [1, +\infty] \\ J_3^\nabla &= J_2^\nabla \nabla J_3 = [1, +\infty] \nabla [1, 3] = [1, +\infty]\end{aligned}$$

- In fact, the maximal chain size arising with this operator is 4:

$$\emptyset \subseteq [3, 7] \subseteq [3, +\infty] \subseteq [-\infty, +\infty]$$

# Applying Widening to Interval Analysis

## Worklist Algorithm with Widening I

**Goal:** extend Algorithm 5.1 by widening to ensure termination

### Algorithm 7.5 (Worklist algorithm with widening)

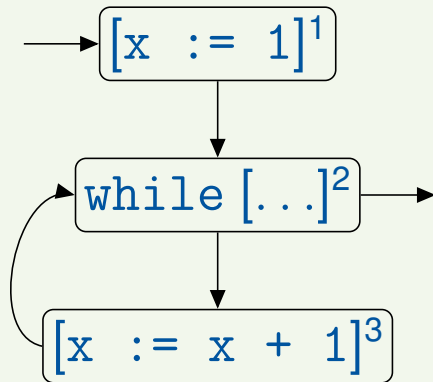
Input: *dataflow system*  $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$   
Variables:  $W \in (Lab \times Lab)^*$ ,  $\{AI_I \in D \mid I \in Lab\}$   
Procedure:  $W := \varepsilon$ ; **for**  $(I, I') \in F$  **do**  $W := W \cdot (I, I')$ ;      % Initialise  $W$   
    **for**  $I \in Lab$  **do**  
        **if**  $I \in E$  **then**  $AI_I := \iota$  **else**  $AI_I := \perp_D$ ;      % Initialise  $AI$   
    **while**  $W \neq \varepsilon$  **do**  
         $(I, I') := \text{head}(W)$ ;  $W := \text{tail}(W)$ ;      % Next control-flow edge  
        **if**  $\varphi_I(AI_I) \not\sqsubseteq AI_{I'}$  **then**      % Fixpoint not yet reached  
             $AI_{I'} := AI_{I'} \nabla \varphi_I(AI_I)$ ;      % Update analysis information  
            **for**  $(I', I'') \in F$  **do**  
                **if**  $(I', I'')$  not in  $W$  **then**  $W := (I', I'') \cdot W$ ;      % Propagate modification  
Output:  $\{AI_I \mid I \in Lab\}$ , denoted by  $\text{fix}^\nabla(\Phi_S)$

**Remark:** due to widening, only  $\text{fix}^\nabla(\Phi_S) \sqsupseteq \text{fix}(\Phi_S)$  is guaranteed (cf. Thm. 5.4)

# Applying Widening to Interval Analysis

## Worklist Algorithm with Widening II

### Example 7.6



Transfer functions (for  $\delta(\mathbf{x}) = \mathcal{J}$ ):

$$\varphi_1(\mathcal{J}) = [1, 1]$$

$$\varphi_2(\mathcal{J}) = \mathcal{J}$$

$$\varphi_3(\emptyset) = \emptyset$$

$$\varphi_3([x_1, x_2]) = [x_1 + 1, x_2 + 1]$$

Application of worklist algorithm with widening:  
terminates with expected result for  $AI_2$  ( $[1, +\infty]$ ) (on the board)