

Automaton-based Array Initialization Analysis

Đurica Nikolić and Fausto Spoto

Dipartimento di Informatica, University of Verona (Italy)

March 9th, 2012



MOTIVATING EXAMPLE - from the CubeWallpaper Android program by Google

```
private void readModel(String prefix) {
    String[] p = ....
    int numpoints = p.length;
    this.mOriginal = new ThreeDPoint[numpoints];
    this.mRotated = new ThreeDPoint[numpoints];
    for (int i = 0; i < numpoints; i++) {
        this.mOriginal[i] = new ThreeDPoint();
        this.mRotated[i] = new ThreeDPoint();
        String[] coord = p[i].split(" ");
        this.mOriginal[i].x=Float.valueOf(coord[0]);
        this.mOriginal[i].y=Float.valueOf(coord[1]);
        this.mOriginal[i].z=Float.valueOf(coord[2]);
    }
    [point *]
}
```

MOTIVATING EXAMPLE - from the CubeWallpaper Android program by Google

```
private void readModel(String prefix) {
    String[] p = ....
    int numpoints = p.length;
    this.mOriginal = new ThreeDPoint[numpoints];
    this.mRotated = new ThreeDPoint[numpoints];
    for (int i = 0; i < numpoints; i++) {
        this.mOriginal[i] = new ThreeDPoint();
        this.mRotated[i] = new ThreeDPoint();
        String[] coord = p[i].split(" ");
        this.mOriginal[i].x=Float.valueOf(coord[0]);
        this.mOriginal[i].y=Float.valueOf(coord[1]);
        this.mOriginal[i].z=Float.valueOf(coord[2]);
    }
    [point *]
}
```

MOTIVATING EXAMPLE - from the CubeWallpaper Android program by Google

```
private void readModel(String prefix) {
    String[] p = ....
    int numpoints = p.length;
    this.mOriginal = new ThreeDPoint[numpoints];
    this.mRotated = new ThreeDPoint[numpoints];
    for (int i = 0; i < numpoints; i++) {
        this.mOriginal[i] = new ThreeDPoint();
        this.mRotated[i] = new ThreeDPoint();
        String[] coord = p[i].split(" ");
        this.mOriginal[i].x=Float.valueOf(coord[0]);
        this.mOriginal[i].y=Float.valueOf(coord[1]);
        this.mOriginal[i].z=Float.valueOf(coord[2]);
    }
    [point *]
}
```

mOriginal and *mRotated* are COMPLETELY INITIALIZED at [point *]

MOTIVATING EXAMPLE - from the CubeWallpaper Android program by Google

```
private void readModel(String prefix) {
    String[] p = ....
    int numpoints = p.length;
    this.mOriginal = new ThreeDPoint[numpoints];
    this.mRotated = new ThreeDPoint[numpoints];
    for (int i = 0; i < numpoints; i++) {
        this.mOriginal[i] = new ThreeDPoint();
        this.mRotated[i] = new ThreeDPoint();
        String[] coord = p[i].split(" ");
        this.mOriginal[i].x=Float.valueOf(coord[0]);
        this.mOriginal[i].y=Float.valueOf(coord[1]);
        this.mOriginal[i].z=Float.valueOf(coord[2]);
    }
    [point *]
}
```

Our nullness analysis
used to consider
elements of
mOriginal and
mRotated
POTENTIALLY NULL at
[point *]

mOriginal and *mRotated* are **COMPLETELY INITIALIZED** at [point *]

RELATED WORK

- Complete initialization of arrays to some value is UNDECIDABLE
- STATIC ANALYSIS can often help
- the majority of existing techniques ARE NOT AUTOMATIC
- the most precise existing approach is [CousotCousotLogozzo2011]
 - our approach is SIMPLER and FASTER, but LESS PRECISE
 - unlike theirs, our approach has been FORMALLY PROVEN CORRECT

GOAL: DEFINE, FORMALLY PROVE CORRECT AND IMPLEMENT AN ARRAY INITIALIZATION ANALYSIS FOR A JAVA-LIKE LANGUAGE

- 1 define SYNTAX and OPERATIONAL SEMANTICS of a Java-like language
- 2 define an ABSTRACT INTERPRETATION of the operational semantics
- 3 PROVE 1 and 2 related by a correctness relation
- 4 provide a STATIC ANALYSIS ALGORITHM
- 5 EXPERIMENTAL EVALUATION of the approach
- 6 extension of analysis to MULTI-DIMENSIONAL ARRAYS

SYNTAX OF A SIMPLE IMPERATIVE LANGUAGE

| | | |
|---|--|---------------------------|
| $E ::= n$ x $x.length$ $x[E]$ $E \oplus E$ | INTEGER VARIABLE ARRAY LENGTH ARRAY ELEMENT $\oplus \in \{+, -, *, \div, \%\}$ | ARITHMETIC EXPRESSIONS |
| $B ::= true$ $false$ $\neg B$ $E \otimes E$ $B \odot B$ | TRUTH FALSITY NEGATION $\otimes \in \{<, \leq, =\}$ $\odot \in \{\wedge, \vee\}$ | BOOLEAN EXPRESSIONS |
| $A ::= B$ $x := E$ $x[E] := E$ $x := new\ t[E]$ | TEST VARIABLE ASSIGNMENT ARRAY ELEMENT ASSIGNMENT CREATION OF AN ARRAY | ACTIONS |
| $C ::= L_1 : A \rightarrow L_2;$ | COMMAND | |

EXAMPLE

A JAVA LOOP

```
1. i = 0;
2. if (b<5) {
3.   a[i] = 2;
   }
4. while (i< a.length) {
5.   a[i] = 3;
6.   i++;
   }
7. ...
```

CORRESPONDING TRANSITION SYSTEM

EXAMPLE

A JAVA LOOP

```

1. i = 0;
2. if (b<5) {
3.   a[i] = 2;
   }
4. while (i< a.length) {
5.   a[i] = 3;
6.   i++;
   }
7. ...

```

CORRESPONDING TRANSITION SYSTEM

$$C_0 \quad 1 : i := 0 \rightarrow 2;$$

EXAMPLE

A JAVA LOOP

```

1. i = 0;
2. if (b<5) {
3.   a[i] = 2;
   }
4. while (i< a.length) {
5.   a[i] = 3;
6.   i++;
   }
7. ...

```

CORRESPONDING TRANSITION SYSTEM

```

C0  1 :  i := 0 → 2;
C1  2 :  b < 5 → 3;
C2  2 :  ¬(b < 5) → 4;

```

EXAMPLE

A JAVA LOOP

```

1. i = 0;
2. if (b<5) {
3.   a[i] = 2;
   }
4. while (i< a.length) {
5.   a[i] = 3;
6.   i++;
   }
7. ...

```

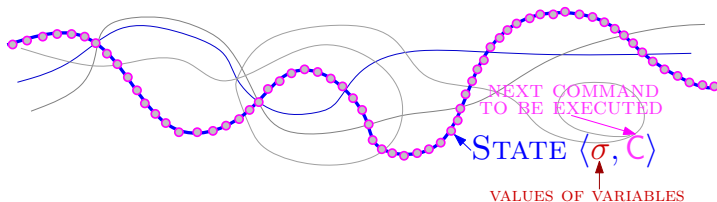
CORRESPONDING TRANSITION SYSTEM

```

C0 1 : i := 0 → 2;
C1 2 : b < 5 → 3;
C2 2 : ¬(b < 5) → 4;
C3 3 : a[i] := 2 → 4;
C4 4 : i < a.length → 5;
C5 4 : ¬(i < a.length) → 7;
C6 5 : a[i] := 3 → 6;
C7 6 : i := i + 1 → 4;
C8 7 : ...

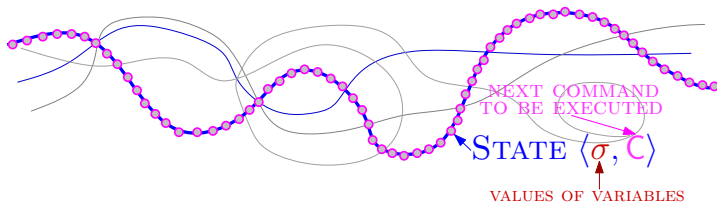
```


TRACE-BASED OPERATIONAL SEMANTICS



- States describe the current CONFIGURATION OF THE SYSTEM

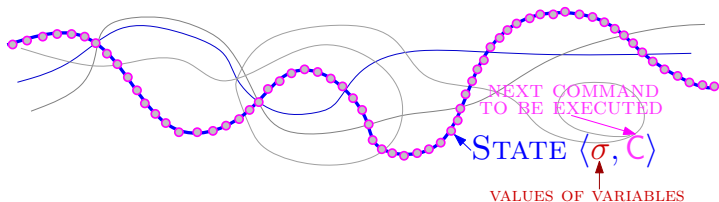
TRACE-BASED OPERATIONAL SEMANTICS



- States describe the current **CONFIGURATION OF THE SYSTEM**

- Operational semantics at C_n : **@ C_n - ALL OF THE TRACES THAT REACH C_n**

TRACE-BASED OPERATIONAL SEMANTICS



- States describe the current **CONFIGURATION OF THE SYSTEM**
- Operational semantics at C_n : **@ C_n - ALL OF THE TRACES THAT REACH C_n**
- Problem: **INFINITE TRACES**

ABSTRACT INTERPRETATION

- To reduce the size of the system, we use **ABSTRACT INTERPRETATION**
- General idea: **ABSTRACT AWAY IRRELEVANT INFORMATION**
- **CONCRETE STATE: $\langle \sigma, C \rangle$ vs. ABSTRACT STATE: $\langle \sigma, \alpha(C) \rangle$,**
i.e., abstract states are obtained from concrete states by removing variables' values and some irrelevant commands

ABSTRACT INTERPRETATION

- To reduce the size of the system, we use **ABSTRACT INTERPRETATION**
- General idea: **ABSTRACT AWAY IRRELEVANT INFORMATION**
- **CONCRETE STATE**: $\langle \sigma, C \rangle$ vs. **ABSTRACT STATE**: $\langle \sigma, \alpha(C) \rangle$,
i.e., *abstract states are obtained from concrete states by removing variables' values and some irrelevant commands*
- How do we determine **RELEVANT COMMANDS**?

HOW DO WE DETERMINE RELEVANT COMMANDS?

ARRAY OF INTEREST: *a*

INDEX VARIABLE: *i*

```
...  
i = 0;  
...  
while i < a.length do{  
    ...  
    a[i] = ...;  
    ...  
    i ++;  
    ...  
}  
...
```

HOW DO WE DETERMINE RELEVANT COMMANDS?

ARRAY OF INTEREST: *a*

INDEX VARIABLE: *i*

```
...  
i = 0;  
...  
while i < a.length do{  
  ...  
  a[i] = ...;  
  ...  
  i ++;  
  ...  
}  
...
```

ABSTRACTION: 0

INDEX VARIABLE *i* INITIALIZED TO 0

HOW DO WE DETERMINE RELEVANT COMMANDS?

ARRAY OF INTEREST: *a*

INDEX VARIABLE: *i*

```
...  
i = 0;  
...  
while i < a.length do{  
  ...  
  a[i] = ...;  
  ...  
  i ++;  
  ...  
}  
...
```

ABSTRACTION: =
ARRAY ELEMENT **a[i]** INITIALIZED

HOW DO WE DETERMINE RELEVANT COMMANDS?

ARRAY OF INTEREST: *a*

INDEX VARIABLE: *i*

```
...  
i = 0;  
...  
while i < a.length do{  
  ...  
  a[i] = ...;  
  ...  
  i ++;  
  ...  
}  
...
```

ABSTRACTION: +
INDEX VARIABLE *i* INCREMENTED BY 1

HOW DO WE DETERMINE RELEVANT COMMANDS?

ARRAY OF INTEREST: *a*

INDEX VARIABLE: *i*

```
...
i = 0;
...
while i < a.length do{
  ...
  a[i] = ...;
  ...
  i ++;
  ...
}
```

ABSTRACTION: \geq

INDEX VARIABLE COMPARED WITH *a.length*

ABSTRACT INTERPRETATION

- To reduce the size of the system, we use **ABSTRACT INTERPRETATION**
- General idea: **ABSTRACT AWAY IRRELEVANT INFORMATION**
- **CONCRETE STATE: $\langle \sigma, C \rangle$ vs. ABSTRACT STATE: $\langle \sigma', \alpha(C) \rangle$,**
i.e., *abstract states are obtained from concrete states by removing variables' values and some irrelevant commands*
- How do we determine **RELEVANT COMMANDS?**

$$\alpha(\langle L_1 : A \rightarrow L_2; \rangle) = \begin{cases} 0 & \text{if } A \text{ is } i := 0 \\ + & \text{if } A \text{ is } i := i + 1 \\ \geq & \text{if } A \text{ is } \neg(i < a.length) \\ = & \text{if } A \text{ is } a[i] = E \\ \mathcal{I} & \text{otherwise, if } \text{mod}(A) \cap \{a, i\} = \emptyset \\ \mathcal{R} & \text{otherwise,} \end{cases}$$

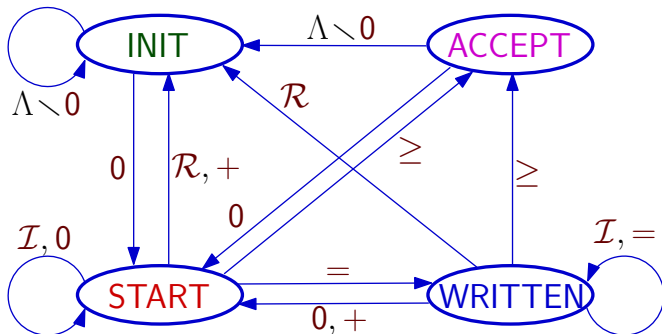
ABSTRACTION OF TRACES

We abstract traces by **REGULAR EXPRESSIONS** on $\Lambda = \{0, =, +, \geq, \mathcal{I}, \mathcal{R}\}$

$$\begin{array}{ll}
 \text{INIT} & \equiv \Lambda^* \\
 \text{START} & \equiv \Lambda^* 0 \mathcal{I}^* ((=\mathcal{I}^*)^+ + \mathcal{I}^*)^* \\
 \text{WRITTEN} & \equiv \Lambda^* 0 \mathcal{I}^* ((=\mathcal{I}^*)^+ + \mathcal{I}^*)^* (=\mathcal{I}^*)^+ \\
 \text{ACCEPT} & \equiv \Lambda^* 0 \mathcal{I}^* ((=\mathcal{I}^*)^+ + \mathcal{I}^*)^* (=\mathcal{I}^*)^* \geq
 \end{array}$$

ABSTRACT SEMANTICS: FINITE-STATE DETERMINISTIC AUTOMATON

- **ALPHABET** $\Lambda = \{0, +, =, \geq, \mathcal{I}, \mathcal{R}\}$
- **STATES** $S = \{\text{INIT}, \text{START}, \text{WRITTEN}, \text{ACCEPT}\}$;
- **TRANSITION FUNCTION** $\delta : S \times \Lambda \rightarrow S$: given states $p, q \in S$ and $\lambda \in \Lambda$, if the automaton has a transition from p to q labelled by λ , then $\delta(p, \lambda) = q$.



THE STATIC ANALYSIS ALGORITHM: ARRAYINIT

Algorithm for a specific pairs $\langle a, i \rangle$

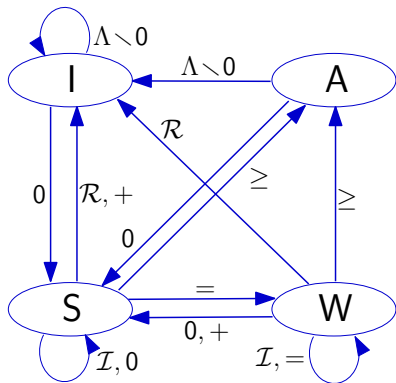
```

1: for all  $C \in \mathbb{C}$  do
2:    $\varphi(C) := \emptyset$ ;
3: end for
4:  $ws := [\langle C_{init}, INIT \rangle]$ ;
5:  $\varphi(C_{init}) := \{INIT\}$ ;
6: while ( $!ws.isEmpty()$ ) do
7:    $\langle C, \sigma^\# \rangle := ws.pop()$ ;
8:   for all  $C_1$  such that  $suc(C) = ini(C_1)$  do
9:      $\sigma_1^\# := \delta(\sigma^\#, s(C))$ ;
10:    if ( $\sigma_1^\# \notin \varphi(C_1)$ ) then
11:       $ws.push(\langle C_1, \sigma_1^\# \rangle)$ ;
12:       $\varphi(C_1) := \varphi(C_1) \cup \{\sigma_1^\#\}$ ;
13:    end if
14:  end for
15: end while

```

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

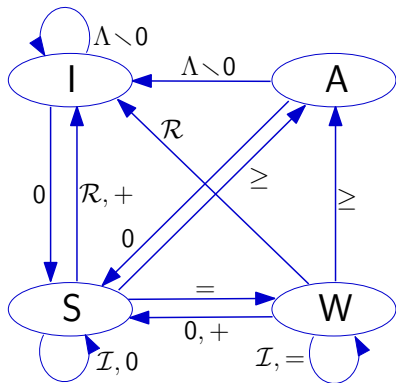
| | | | | | |
|---|----------------|---|---|----------------------|------|
| I | C ₀ | 1 | : | $i := 0$ | → 2; |
| ∅ | C ₁ | 2 | : | $b < 5$ | → 3; |
| ∅ | C ₂ | 2 | : | $\neg(b < 5)$ | → 4; |
| ∅ | C ₃ | 3 | : | $a[i] := 2$ | → 4; |
| ∅ | C ₄ | 4 | : | $i < a.length$ | → 5; |
| ∅ | C ₅ | 4 | : | $\neg(i < a.length)$ | → 7; |
| ∅ | C ₆ | 5 | : | $a[i] := 3$ | → 6; |
| ∅ | C ₇ | 6 | : | $i := i + 1$ | → 4; |
| | C ₈ | 7 | : | ... | |



EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

COMMANDS

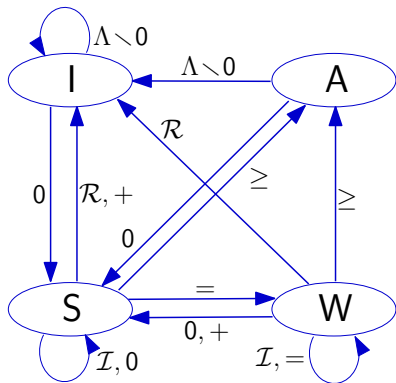
| | | | |
|-------------|-------|--------------------------|-------------------|
| I | C_0 | 1 : $i := 0$ | $\rightarrow 2$; |
| \emptyset | C_1 | 2 : $b < 5$ | $\rightarrow 3$; |
| \emptyset | C_2 | 2 : $\neg(b < 5)$ | $\rightarrow 4$; |
| \emptyset | C_3 | 3 : $a[i] := 2$ | $\rightarrow 4$; |
| \emptyset | C_4 | 4 : $i < a.length$ | $\rightarrow 5$; |
| \emptyset | C_5 | 4 : $\neg(i < a.length)$ | $\rightarrow 7$; |
| \emptyset | C_6 | 5 : $a[i] := 3$ | $\rightarrow 6$; |
| \emptyset | C_7 | 6 : $i := i + 1$ | $\rightarrow 4$; |
| \emptyset | C_8 | 7 : ... | |



EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

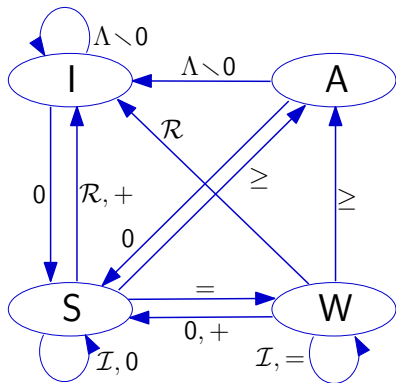
REACHABLE STATES

| | | | | | |
|---|----------------|---|---|----------------------|-------------------|
| I | C ₀ | 1 | : | $i := 0$ | $\rightarrow 2$; |
| ∅ | C ₁ | 2 | : | $b < 5$ | $\rightarrow 3$; |
| ∅ | C ₂ | 2 | : | $\neg(b < 5)$ | $\rightarrow 4$; |
| ∅ | C ₃ | 3 | : | $a[i] := 2$ | $\rightarrow 4$; |
| ∅ | C ₄ | 4 | : | $i < a.length$ | $\rightarrow 5$; |
| ∅ | C ₅ | 4 | : | $\neg(i < a.length)$ | $\rightarrow 7$; |
| ∅ | C ₆ | 5 | : | $a[i] := 3$ | $\rightarrow 6$; |
| ∅ | C ₇ | 6 | : | $i := i + 1$ | $\rightarrow 4$; |
| | C ₈ | 7 | : | ... | |



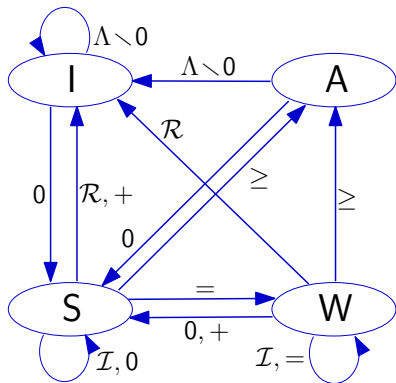
EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | | | |
|---|----------------|---|---|----------------------|------|
| I | C ₀ | 1 | : | $i := 0$ | → 2; |
| ∅ | C ₁ | 2 | : | $b < 5$ | → 3; |
| ∅ | C ₂ | 2 | : | $\neg(b < 5)$ | → 4; |
| ∅ | C ₃ | 3 | : | $a[i] := 2$ | → 4; |
| ∅ | C ₄ | 4 | : | $i < a.length$ | → 5; |
| ∅ | C ₅ | 4 | : | $\neg(i < a.length)$ | → 7; |
| ∅ | C ₆ | 5 | : | $a[i] := 3$ | → 6; |
| ∅ | C ₇ | 6 | : | $i := i + 1$ | → 4; |
| | C ₈ | 7 | : | ... | |



EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

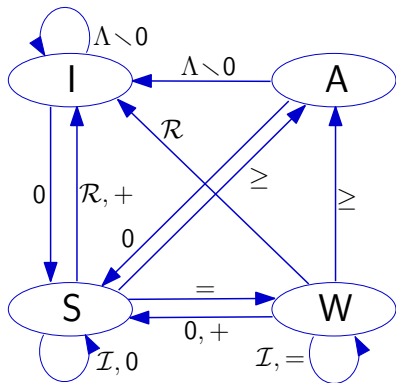
| | | | | |
|---|----------------|---|------------------------|------|
| I | C ₀ | 1 | : $i := 0$ | → 2; |
| ∅ | C ₁ | 2 | : $b < 5$ | → 3; |
| ∅ | C ₂ | 2 | : $\neg(b < 5)$ | → 4; |
| ∅ | C ₃ | 3 | : $a[i] := 2$ | → 4; |
| ∅ | C ₄ | 4 | : $i < a.length$ | → 5; |
| ∅ | C ₅ | 4 | : $\neg(i < a.length)$ | → 7; |
| ∅ | C ₆ | 5 | : $a[i] := 3$ | → 6; |
| ∅ | C ₇ | 6 | : $i := i + 1$ | → 4; |
| | C ₈ | 7 | : ... | |



EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | | |
|---|----------------|-----|--|-------|
| I | C ₀ | 1 : | $i := 0$ | → 2 ; |
| ∅ | C ₁ | 2 : | $b < 5$ | → 3 ; |
| ∅ | C ₂ | 2 : | $\neg(b < 5)$ | → 4 ; |
| ∅ | C ₃ | 3 : | $a[i] := 2$ | → 4 ; |
| ∅ | C ₄ | 4 : | $i < a.length$ | → 5 ; |
| ∅ | C ₅ | 4 : | $\neg(i < a.length)$ | → 7 ; |
| ∅ | C ₆ | 5 : | $a[i] := 3$ | → 6 ; |
| ∅ | C ₇ | 6 : | $i := i + 1$ | → 4 ; |
| ∅ | C ₈ | 7 : | ... | |

ABSTRACTION OF COMMAND: 0

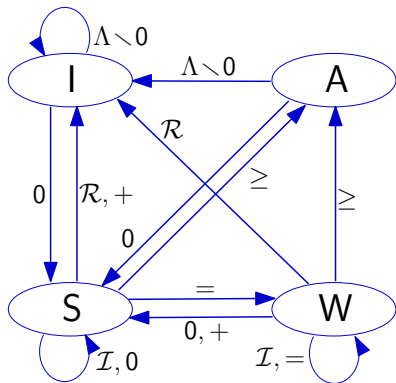


EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | | |
|---|----------------|-----|----------------------|-------|
| I | C ₀ | 1 : | $i := 0$ | → 2 ; |
| ∅ | C ₁ | 2 : | $b < 5$ | → 3 ; |
| ∅ | C ₂ | 2 : | $\neg(b < 5)$ | → 4 ; |
| ∅ | C ₃ | 3 : | $a[i] := 2$ | → 4 ; |
| ∅ | C ₄ | 4 : | $i < a.length$ | → 5 ; |
| ∅ | C ₅ | 4 : | $\neg(i < a.length)$ | → 7 ; |
| ∅ | C ₆ | 5 : | $a[i] := 3$ | → 6 ; |
| ∅ | C ₇ | 6 : | $i := i + 1$ | → 4 ; |
| ∅ | C ₈ | 7 : | ... | |

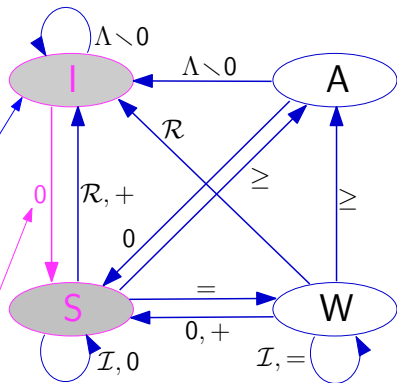
ABSTRACTION OF COMMAND: 0

CURRENT STATE: I



EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | |
|---|----------------|--------------------------|-------|
| I | C ₀ | 1 : $i := 0$ | → 2 ; |
| ∅ | C ₁ | 2 : $b < 5$ | → 3 ; |
| ∅ | C ₂ | 2 : $\neg(b < 5)$ | → 4 ; |
| ∅ | C ₃ | 3 : $a[i] := 2$ | → 4 ; |
| ∅ | C ₄ | 4 : $i < a.length$ | → 5 ; |
| ∅ | C ₅ | 4 : $\neg(i < a.length)$ | → 7 ; |
| ∅ | C ₆ | 5 : $a[i] := 3$ | → 6 ; |
| ∅ | C ₇ | 6 : $i := i + 1$ | → 4 ; |
| ∅ | C ₈ | 7 : ... | |

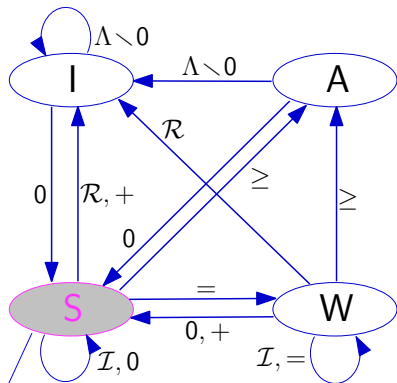


ABSTRACTION OF COMMAND: 0

CURRENT STATE: I

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | |
|---|----------------|--------------------------|-------------------|
| I | C ₀ | 1 : $i := 0$ | $\rightarrow 2$; |
| ∅ | C ₁ | 2 : $b < 5$ | $\rightarrow 3$; |
| ∅ | C ₂ | 2 : $\neg(b < 5)$ | $\rightarrow 4$; |
| ∅ | C ₃ | 3 : $a[i] := 2$ | $\rightarrow 4$; |
| ∅ | C ₄ | 4 : $i < a.length$ | $\rightarrow 5$; |
| ∅ | C ₅ | 4 : $\neg(i < a.length)$ | $\rightarrow 7$; |
| ∅ | C ₆ | 5 : $a[i] := 3$ | $\rightarrow 6$; |
| ∅ | C ₇ | 6 : $i := i + 1$ | $\rightarrow 4$; |
| | C ₈ | 7 : ... | |



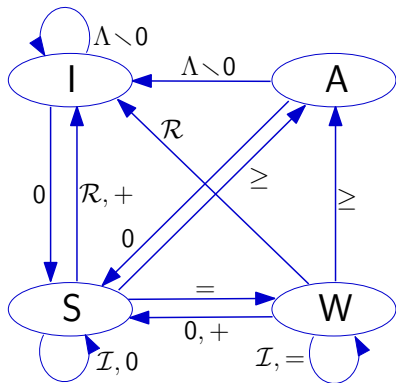
ABSTRACTION OF COMMAND: 0

CURRENT STATE: I NEXT STATE: S

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

NEXT COMMAND(S)

| | | | |
|---|----------------|--------------------------|------|
| I | C ₀ | 1 : $i := 0$ | → 2; |
| ∅ | C ₁ | 2 : $b < 5$ | → 3; |
| ∅ | C ₂ | 2 : $\neg(b < 5)$ | → 4; |
| ∅ | C ₃ | 3 : $a[i] := 2$ | → 4; |
| ∅ | C ₄ | 4 : $i < a.length$ | → 5; |
| ∅ | C ₅ | 4 : $\neg(i < a.length)$ | → 7; |
| ∅ | C ₆ | 5 : $a[i] := 3$ | → 6; |
| ∅ | C ₇ | 6 : $i := i + 1$ | → 4; |
| | C ₈ | 7 : ... | |

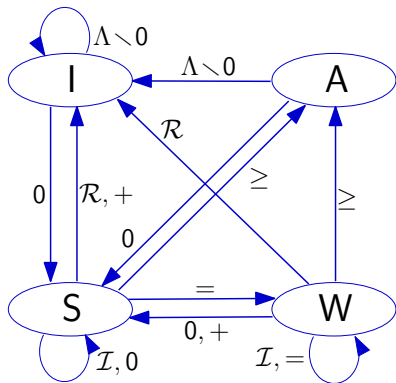


ABSTRACTION OF COMMAND: 0

CURRENT STATE: I NEXT STATE: S

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | | | |
|---|----------------|---|----------------------|---|----|
| I | C ₀ | 1 | $i := 0$ | → | 2; |
| S | C ₁ | 2 | $b < 5$ | → | 3; |
| S | C ₂ | 2 | $\neg(b < 5)$ | → | 4; |
| ∅ | C ₃ | 3 | $a[i] := 2$ | → | 4; |
| ∅ | C ₄ | 4 | $i < a.length$ | → | 5; |
| ∅ | C ₅ | 4 | $\neg(i < a.length)$ | → | 7; |
| ∅ | C ₆ | 5 | $a[i] := 3$ | → | 6; |
| ∅ | C ₇ | 6 | $i := i + 1$ | → | 4; |
| | C ₈ | 7 | ... | | |

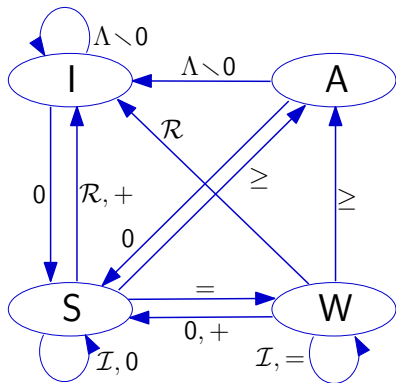


ABSTRACTION OF COMMAND: 0

CURRENT STATE: I NEXT STATE: S

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

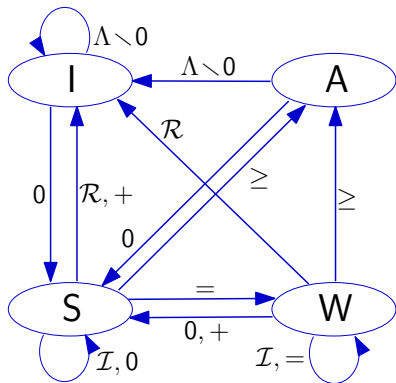
| | | | | | |
|---|----------------|---|---|----------------------|------|
| I | C ₀ | 1 | : | $i := 0$ | → 2; |
| S | C ₁ | 2 | : | $b < 5$ | → 3; |
| S | C ₂ | 2 | : | $\neg(b < 5)$ | → 4; |
| ∅ | C ₃ | 3 | : | $a[i] := 2$ | → 4; |
| ∅ | C ₄ | 4 | : | $i < a.length$ | → 5; |
| ∅ | C ₅ | 4 | : | $\neg(i < a.length)$ | → 7; |
| ∅ | C ₆ | 5 | : | $a[i] := 3$ | → 6; |
| ∅ | C ₇ | 6 | : | $i := i + 1$ | → 4; |
| | C ₈ | 7 | : | ... | |



WORKING SET: $\langle C_2, S \rangle$, $\langle C_1, S \rangle$

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

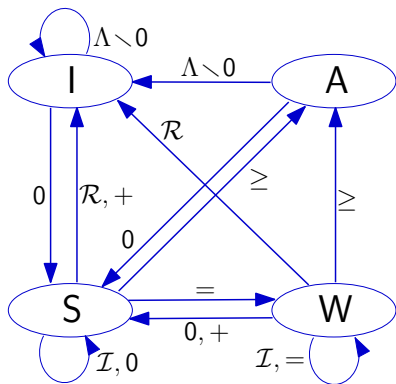
| | | | | | |
|---|----------------|---|---|----------------------|------|
| I | C ₀ | 1 | : | $i := 0$ | → 2; |
| S | C ₁ | 2 | : | $b < 5$ | → 3; |
| S | C ₂ | 2 | : | $\neg(b < 5)$ | → 4; |
| ∅ | C ₃ | 3 | : | $a[i] := 2$ | → 4; |
| ∅ | C ₄ | 4 | : | $i < a.length$ | → 5; |
| ∅ | C ₅ | 4 | : | $\neg(i < a.length)$ | → 7; |
| ∅ | C ₆ | 5 | : | $a[i] := 3$ | → 6; |
| ∅ | C ₇ | 6 | : | $i := i + 1$ | → 4; |
| | C ₈ | 7 | : | ... | |



WORKING SET: $\langle C_2, S \rangle$, $\langle C_1, S \rangle$

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

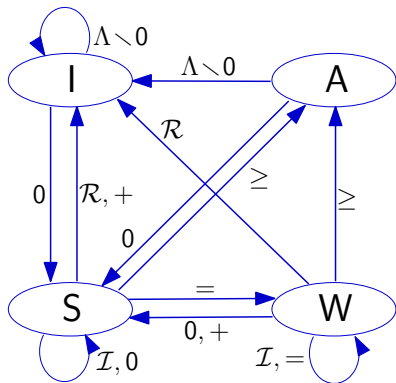
| | | | | | |
|---|----------------|---|---|----------------------|------|
| I | C ₀ | 1 | : | $i := 0$ | → 2; |
| S | C ₁ | 2 | : | $b < 5$ | → 3; |
| S | C ₂ | 2 | : | $\neg(b < 5)$ | → 4; |
| ∅ | C ₃ | 3 | : | $a[i] := 2$ | → 4; |
| ∅ | C ₄ | 4 | : | $i < a.length$ | → 5; |
| ∅ | C ₅ | 4 | : | $\neg(i < a.length)$ | → 7; |
| ∅ | C ₆ | 5 | : | $a[i] := 3$ | → 6; |
| ∅ | C ₇ | 6 | : | $i := i + 1$ | → 4; |
| | C ₈ | 7 | : | ... | |



WORKING SET: $\langle C_2, S \rangle$, $\langle C_1, S \rangle$

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | | | |
|---|----------------|---|---|----------------------|------|
| I | C ₀ | 1 | : | $i := 0$ | → 2; |
| S | C ₁ | 2 | : | $b < 5$ | → 3; |
| S | C ₂ | 2 | : | $\neg(b < 5)$ | → 4; |
| ∅ | C ₃ | 3 | : | $a[i] := 2$ | → 4; |
| ∅ | C ₄ | 4 | : | $i < a.length$ | → 5; |
| ∅ | C ₅ | 4 | : | $\neg(i < a.length)$ | → 7; |
| ∅ | C ₆ | 5 | : | $a[i] := 3$ | → 6; |
| ∅ | C ₇ | 6 | : | $i := i + 1$ | → 4; |
| | C ₈ | 7 | : | ... | |

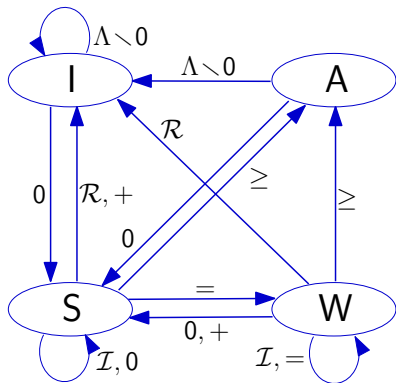


ABSTRACTION OF COMMAND: \mathcal{I}

WORKING SET: $\langle C_2, S \rangle$, $\langle C_1, S \rangle$

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | |
|---|----------------|--------------------------|-------|
| I | C ₀ | 1 : $i := 0$ | → 2 ; |
| S | C ₁ | 2 : $b < 5$ | → 3 ; |
| S | C ₂ | 2 : $\neg(b < 5)$ | → 4 ; |
| ∅ | C ₃ | 3 : $a[i] := 2$ | → 4 ; |
| ∅ | C ₄ | 4 : $i < a.length$ | → 5 ; |
| ∅ | C ₅ | 4 : $\neg(i < a.length)$ | → 7 ; |
| ∅ | C ₆ | 5 : $a[i] := 3$ | → 6 ; |
| ∅ | C ₇ | 6 : $i := i + 1$ | → 4 ; |
| ∅ | C ₈ | 7 : ... | |



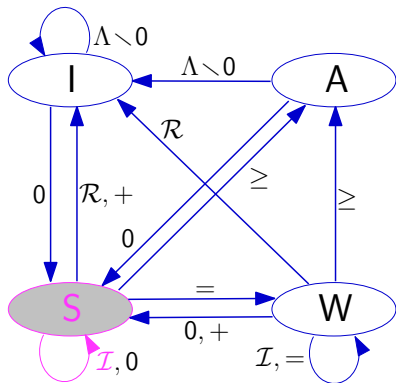
ABSTRACTION OF COMMAND: \mathcal{I}

CURRENT STATE: S

WORKING SET: $\langle C_2, S \rangle$, $\langle C_1, S \rangle$

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | |
|---|----------------|--------------------------|-------|
| I | C ₀ | 1 : $i := 0$ | → 2 ; |
| S | C ₁ | 2 : $b < 5$ | → 3 ; |
| S | C ₂ | 2 : $\neg(b < 5)$ | → 4 ; |
| ∅ | C ₃ | 3 : $a[i] := 2$ | → 4 ; |
| ∅ | C ₄ | 4 : $i < a.length$ | → 5 ; |
| ∅ | C ₅ | 4 : $\neg(i < a.length)$ | → 7 ; |
| ∅ | C ₆ | 5 : $a[i] := 3$ | → 6 ; |
| ∅ | C ₇ | 6 : $i := i + 1$ | → 4 ; |
| ∅ | C ₈ | 7 : ... | |



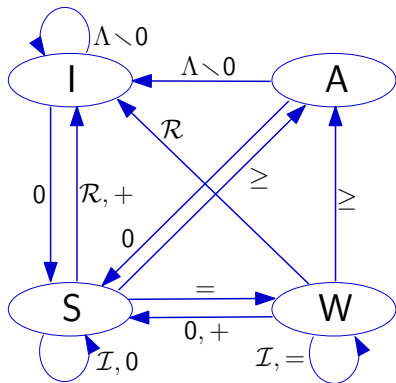
ABSTRACTION OF COMMAND: I

CURRENT STATE: S NEXT STATE: S

WORKING SET: $\langle C_2, S \rangle$, $\langle C_1, S \rangle$

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | | | | |
|---|----------------|---|---|----------------------|---|----|
| I | C ₀ | 1 | : | $i := 0$ | → | 2; |
| S | C ₁ | 2 | : | $b < 5$ | → | 3; |
| S | C ₂ | 2 | : | $\neg(b < 5)$ | → | 4; |
| ∅ | C ₃ | 3 | : | $a[i] := 2$ | → | 4; |
| ∅ | C ₄ | 4 | : | $i < a.length$ | → | 5; |
| ∅ | C ₅ | 4 | : | $\neg(i < a.length)$ | → | 7; |
| ∅ | C ₆ | 5 | : | $a[i] := 3$ | → | 6; |
| ∅ | C ₇ | 6 | : | $i := i + 1$ | → | 4; |
| | C ₈ | 7 | : | ... | | |



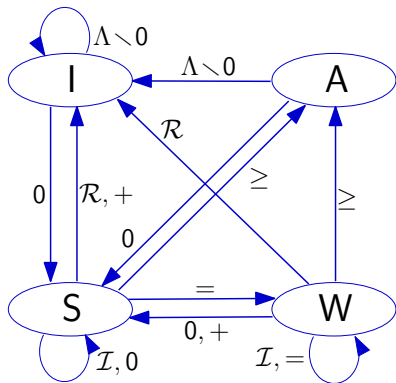
ABSTRACTION OF COMMAND: \mathcal{I}

CURRENT STATE: S NEXT STATE: S

WORKING SET: $\langle C_2, S \rangle$, $\langle C_1, S \rangle$

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | | | | |
|---|----------------|---|---|----------------------|---|----|
| I | C ₀ | 1 | : | $i := 0$ | → | 2; |
| S | C ₁ | 2 | : | $b < 5$ | → | 3; |
| S | C ₂ | 2 | : | $\neg(b < 5)$ | → | 4; |
| S | C ₃ | 3 | : | $a[i] := 2$ | → | 4; |
| ∅ | C ₄ | 4 | : | $i < a.length$ | → | 5; |
| ∅ | C ₅ | 4 | : | $\neg(i < a.length)$ | → | 7; |
| ∅ | C ₆ | 5 | : | $a[i] := 3$ | → | 6; |
| ∅ | C ₇ | 6 | : | $i := i + 1$ | → | 4; |
| | C ₈ | 7 | : | ... | | |



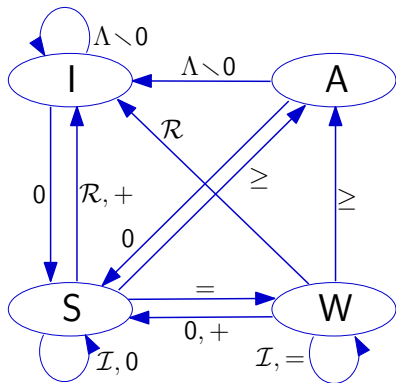
ABSTRACTION OF COMMAND: \mathcal{I}

CURRENT STATE: S NEXT STATE: S

WORKING SET: $\langle C_2, S \rangle$, $\langle C_1, S \rangle$

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

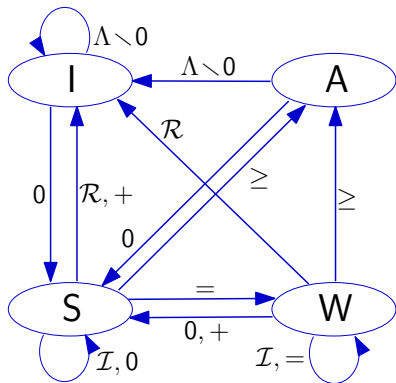
| | | | | | |
|---|----------------|---|---|----------------------|------|
| I | C ₀ | 1 | : | $i := 0$ | → 2; |
| S | C ₁ | 2 | : | $b < 5$ | → 3; |
| S | C ₂ | 2 | : | $\neg(b < 5)$ | → 4; |
| S | C ₃ | 3 | : | $a[i] := 2$ | → 4; |
| ∅ | C ₄ | 4 | : | $i < a.length$ | → 5; |
| ∅ | C ₅ | 4 | : | $\neg(i < a.length)$ | → 7; |
| ∅ | C ₆ | 5 | : | $a[i] := 3$ | → 6; |
| ∅ | C ₇ | 6 | : | $i := i + 1$ | → 4; |
| | C ₈ | 7 | : | ... | |



WORKING SET: $\langle C_3, S \rangle$, $\langle C_2, S \rangle$, $\langle C_1, S \rangle$

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

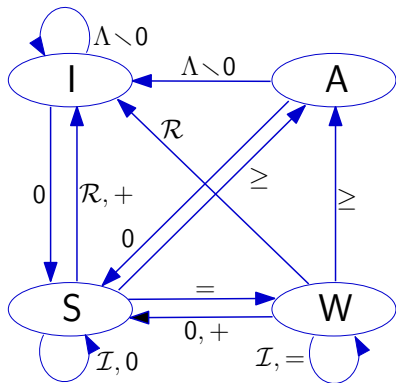
| | | | | | |
|---|----------------|---|---|----------------------|------|
| I | C ₀ | 1 | : | $i := 0$ | → 2; |
| S | C ₁ | 2 | : | $b < 5$ | → 3; |
| S | C ₂ | 2 | : | $\neg(b < 5)$ | → 4; |
| S | C ₃ | 3 | : | $a[i] := 2$ | → 4; |
| ∅ | C ₄ | 4 | : | $i < a.length$ | → 5; |
| ∅ | C ₅ | 4 | : | $\neg(i < a.length)$ | → 7; |
| ∅ | C ₆ | 5 | : | $a[i] := 3$ | → 6; |
| ∅ | C ₇ | 6 | : | $i := i + 1$ | → 4; |
| | C ₈ | 7 | : | ... | |



WORKING SET: $\langle C_3, S \rangle$, $\langle C_2, S \rangle$

EXAMPLE OF APPLICATION OF THE ARRAYINIT ALGORITHM

| | | | | | |
|------|----------------|---|---|----------------------|------|
| I | C ₀ | 1 | : | $i := 0$ | → 2; |
| S | C ₁ | 2 | : | $b < 5$ | → 3; |
| S | C ₂ | 2 | : | $\neg(b < 5)$ | → 4; |
| S | C ₃ | 3 | : | $a[i] := 2$ | → 4; |
| S, W | C ₄ | 4 | : | $i < a.length$ | → 5; |
| S, W | C ₅ | 4 | : | $\neg(i < a.length)$ | → 7; |
| S, W | C ₆ | 5 | : | $a[i] := 3$ | → 6; |
| W | C ₇ | 6 | : | $i := i + 1$ | → 4; |
| A | C ₈ | 7 | : | ... | |



WORKING SET: \emptyset - **FIXPOINT REACHED**

SOUNDNESS OF THE ARRAYINIT ALGORITHM

Lemma

At the end of `ARRAYINIT`, $@C \subseteq \cup \varphi(C)$ holds for each command `C`.

Theorem

Consider a program P , variables a (`ARRAY`) and i (`INDEX`) and the automaton for a and i . At the end of the `ARRAYINIT` algorithm, for each command `C` such that $\varphi(C) = \{\text{ACCEPT}\}$, `C` IS A POINT OF P WHERE ALL ELEMENTS OF a HAVE BEEN INITIALIZED BY A LOOP WITH INDEX i .

JULIA - A STATIC ANALYZER FOR JAVA AND ANDROID



- Julia analyzer **FINDS BUGS IN JAVA AND ANDROID PROGRAMS** well **BEFORE THEY ARE RUN**. It is a **SEMANTICAL TOOL**, based on abstract interpretation, which checks all possible executions of a software and finds all possible bugs, inside the categories considered by the tool.
- Julia is very simple to use: it requires one or more **JAR FILES** making up the software, and it performs the following analyses:
 - **NULLNESS**: is there a pointer dereferenced before being initialized?

JULIA - A STATIC ANALYZER FOR JAVA AND ANDROID



- Julia analyzer **FINDS BUGS IN JAVA AND ANDROID PROGRAMS** well **BEFORE THEY ARE RUN**. It is a **SEMANTICAL TOOL**, based on abstract interpretation, which checks all possible executions of a software and finds all possible bugs, inside the categories considered by the tool.

- Julia is very simple to use: it requires one or more **JAR FILES** making up the software, and it performs the following analyses:

- **NULLNESS**: is there a pointer dereferenced before being initialized?

ARRAYINIT has been implemented inside Julia.

EXPERIMENTAL EVALUATION

| NAME | LOC | TOTAL LOC | ARRAYINIT | | | TOTAL TIME |
|-------------------|-------|--------------|-----------|----------|------|---------------|
| | | | TOTAL | DETECTED | TIME | |
| AbdTest | 489 | 56334 | 1 | 1 | 2.36 | 121.73 |
| AccelerometerPlay | 306 | 46854 | 1 | 1 | 0.35 | 71.99 |
| CubeWallpaper | 370 | 25654 | 3 | 3 | 0.12 | 28.51 |
| HoneycombGallery | 948 | 71501 | 1 | 0 | 1.06 | 157.85 |
| TicTacToe | 607 | 59040 | 3 | 3 | 0.70 | 102.65 |
| Snake | 420 | 57075 | 1 | 0 | 0.36 | 117.49 |
| Real3D | 1228 | 74384 | 2 | 2 | 1.06 | 177.95 |
| ChimeTimer | 4095 | 95781 | 9 | 7 | 0.80 | 383.45 |
| Dazzle | 4376 | 100271 | 4 | 1 | 1.02 | 394.44 |
| OnWatch | 9746 | 113368 | 10 | 6 | 2.91 | 525.15 |
| Tricorder | 10410 | 106100 | 17 | 11 | 1.01 | 467.58 |
| TestAppv2 | 377 | 58365 | 1 | 1 | 0.38 | 102.34 |
| TxWthr | 2024 | 74441 | 7 | 1 | 0.42 | 179.78 |
| JFlex | 7681 | 40872 | 7 | 6 | 1.35 | 72.46 |
| nti | 2372 | 13098 | 4 | 4 | 0.09 | 13.55 |
| plume | 8587 | 43302 | 24 | 21 | 1.19 | 113.07 |

EXPERIMENTAL EVALUATION

| NAME | LOC | TOTAL LOC | ARRAYINIT | | | TOTAL TIME |
|-------------------|-------|--------------|-----------|----------|------|---------------|
| | | | TOTAL | DETECTED | TIME | |
| AbdTest | 489 | 56334 | 1 | 1 | 2.36 | 121.73 |
| AccelerometerPlay | 306 | 46854 | 1 | 1 | 0.35 | 71.99 |
| CubeWallpaper | 370 | 25654 | 3 | 3 | 0.12 | 28.51 |
| HoneycombGallery | 948 | 71501 | 1 | 0 | 1.06 | 157.85 |
| TicTacToe | 607 | 59040 | 3 | 3 | 0.70 | 102.65 |
| Snake | 420 | 57075 | 1 | 0 | 0.36 | 117.49 |
| Real3D | 1228 | 74384 | 2 | 2 | 1.06 | 177.95 |
| ChimeTimer | 4095 | 95781 | 9 | 7 | 0.80 | 383.45 |
| Dazzle | 4376 | 100271 | 4 | 1 | 1.02 | 394.44 |
| OnWatch | 9746 | 113368 | 10 | 6 | 2.91 | 525.15 |
| Tricorder | 10410 | 106100 | 17 | 11 | 1.01 | 467.58 |
| TestAppv2 | 377 | 58365 | 1 | 1 | 0.38 | 102.34 |
| TxWthr | 2024 | 74441 | 7 | 1 | 0.42 | 179.78 |
| JFlex | 7681 | 40872 | 7 | 6 | 1.35 | 72.46 |
| nti | 2372 | 13098 | 4 | 4 | 0.09 | 13.55 |
| plume | 8587 | 43302 | 24 | 21 | 1.19 | 113.07 |

EXPERIMENTAL EVALUATION

| NAME | LOC | TOTAL LOC | ARRAYINIT | | | TOTAL TIME |
|-------------------|-------|--------------|-----------|----------|------|---------------|
| | | | TOTAL | DETECTED | TIME | |
| AbdTest | 489 | 56334 | 1 | 1 | 2.36 | 121.73 |
| AccelerometerPlay | 306 | 46854 | 1 | 1 | 0.35 | 71.99 |
| CubeWallpaper | 370 | 25654 | 3 | 3 | 0.12 | 28.51 |
| HoneycombGallery | 948 | 71501 | 1 | 0 | 1.06 | 157.85 |
| TicTacToe | 607 | 59040 | 3 | 3 | 0.70 | 102.65 |
| Snake | 420 | 57075 | 1 | 0 | 0.36 | 117.49 |
| Real3D | 1228 | 74384 | 2 | 2 | 1.06 | 177.95 |
| ChimeTimer | 4095 | 95781 | 9 | 7 | 0.80 | 383.45 |
| Dazzle | 4376 | 100271 | 4 | 1 | 1.02 | 394.44 |
| OnWatch | 9746 | 113368 | 10 | 6 | 2.91 | 525.15 |
| Tricorder | 10410 | 106100 | 17 | 11 | 1.01 | 467.58 |
| TestAppv2 | 377 | 58365 | 1 | 1 | 0.38 | 102.34 |
| TxWthr | 2024 | 74441 | 7 | 1 | 0.42 | 179.78 |
| JFlex | 7681 | 40872 | 7 | 6 | 1.35 | 72.46 |
| nti | 2372 | 13098 | 4 | 4 | 0.09 | 13.55 |
| plume | 8587 | 43302 | 24 | 21 | 1.19 | 113.07 |

EXPERIMENTAL EVALUATION

| NAME | LOC | TOTAL LOC | ARRAYINIT | | | TOTAL TIME |
|-------------------|-------|--------------|-----------|----------|------|---------------|
| | | | TOTAL | DETECTED | TIME | |
| AbdTest | 489 | 56334 | 1 | 1 | 2.36 | 121.73 |
| AccelerometerPlay | 306 | 46854 | 1 | 1 | 0.35 | 71.99 |
| CubeWallpaper | 370 | 25654 | 3 | 3 | 0.12 | 28.51 |
| HoneycombGallery | 948 | 71501 | 1 | 0 | 1.06 | 157.85 |
| TicTacToe | 607 | 59040 | 3 | 3 | 0.70 | 102.65 |
| Snake | 420 | 57075 | 1 | 0 | 0.36 | 117.49 |
| Real3D | 1228 | 74384 | 2 | 2 | 1.06 | 177.95 |
| ChimeTimer | 4095 | 95781 | 9 | 7 | 0.80 | 383.45 |
| Dazzle | 4376 | 100271 | 4 | 1 | 1.02 | 394.44 |
| OnWatch | 9746 | 113368 | 10 | 6 | 2.91 | 525.15 |
| Tricorder | 10410 | 106100 | 17 | 11 | 1.01 | 467.58 |
| TestAppv2 | 377 | 58365 | 1 | 1 | 0.38 | 102.34 |
| TxWthr | 2024 | 74441 | 7 | 1 | 0.42 | 179.78 |
| JFlex | 7681 | 40872 | 7 | 6 | 1.35 | 72.46 |
| nti | 2372 | 13098 | 4 | 4 | 0.09 | 13.55 |
| plume | 8587 | 43302 | 24 | 21 | 1.19 | 113.07 |

ARRAYINIT INCREASES THE TOTAL TIME BY ONLY 0.47%.

EXPERIMENTAL EVALUATION

| NAME | NULLNESS | |
|-------------------|-----------|-----------|
| | ARRAYINIT | ARRAYINIT |
| AccelerometerPlay | 3 | 6 |
| ChimeTimer | 33 | 36 |
| CubeWallpaper | 0 | 3 |
| TicTacToe | 0 | 2 |
| JFlex | 57 | 65 |
| OnWatch | 82 | 85 |
| Real3D | 19 | 19 |
| Tricorder | 107 | 121 |
| TxWthr | 48 | 49 |
| nti | 15 | 15 |
| plume | 57 | 59 |

EXPERIMENTAL EVALUATION

| NAME | NULLNESS | |
|-------------------|-----------|-----------|
| | ARRAYINIT | ARRAYINIT |
| AccelerometerPlay | 3 | 6 |
| ChimeTimer | 33 | 36 |
| CubeWallpaper | 0 | 3 |
| TicTacToe | 0 | 2 |
| JFlex | 57 | 65 |
| OnWatch | 82 | 85 |
| Real3D | 19 | 19 |
| Tricorder | 107 | 121 |
| TxWthr | 48 | 49 |
| nti | 15 | 15 |
| plume | 57 | 59 |

ARRAYINIT IMPROVES THE PRECISION OF THE NULLNESS ANALYSIS BY 8.48% on the average.

GOAL: DEFINE, FORMALLY PROVE CORRECT AND IMPLEMENT AN ARRAY INITIALIZATION ANALYSIS FOR A JAVA-LIKE LANGUAGE

- 1 define SYNTAX and OPERATIONAL SEMANTICS of a Java-like language
- 2 define an ABSTRACT INTERPRETATION of the operational semantics
- 3 PROVE 1 and 2 related by a correctness relation
- 4 provide a STATIC ANALYSIS ALGORITHM
- 5 EXPERIMENTAL EVALUATION of the approach
- 6 extension of analysis to MULTI-DIMENSIONAL ARRAYS

GOAL: DEFINE, FORMALLY PROVE CORRECT AND IMPLEMENT AN ARRAY INITIALIZATION ANALYSIS FOR A JAVA-LIKE LANGUAGE

- 1 define SYNTAX and OPERATIONAL SEMANTICS of a Java-like language **DONE**
- 2 define an ABSTRACT INTERPRETATION of the operational semantics **DONE**
- 3 PROVE 1 and 2 related by a correctness relation **DONE**
- 4 provide a STATIC ANALYSIS ALGORITHM **DONE**
- 5 EXPERIMENTAL EVALUATION of the approach **DONE**
- 6 extension of analysis to MULTI-DIMENSIONAL ARRAYS (submitted for publication) **DONE**

THANK YOU