Complexity Analysis for Java with AProVE

Florian Frohn[1] Jürgen Giesl[1]

[1]RWTH Aachen University, Germany

December 4, 2017

Java and JBC

```
class List{
  int value; List next;
  List(int v, List n){...}
  boolean member(int n){...}
  int max(){...}

  List sort(){
    int n = 0;
    List r = null;
    while (this.max() >= n){
      if (this.member(n))
        r = new List(n,r);
      n++;
    }
    return r;
  }

}
```

Java and JBC

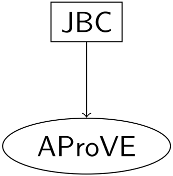
```java
class List{
  int value; List next;
  List(int v, List n){...}
  boolean member(int n){...}
  int max(){...}

  List sort(){
    int n = 0;
    List r = null;
    while (this.max() >= n){
      if (this.member(n))
        r = new List(n,r);
      n++;
    }
    return r;
  }

}
```

```
List sort();
  Code:
     0: iconst_0
     1: istore_1
     2: aconst_null
     3: astore_2
     4: aload_0
     5: invokevirtual #4
     8: iload_1
     9: if_icmplt 36
    12: aload_0
    13: iload_1
    14: invokevirtual #5
    17: ifeq 30
    20: new #6
    23: dup
    24: iload_1
    25: aload_2
    26: invokespecial #7
    29: astore_2
    30: iinc 1, 1
    33: goto 4
    36: aload_2
    37: areturn
```

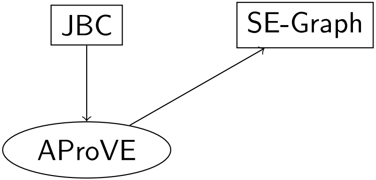
Overview

$\boxed{\text{JBC}}$

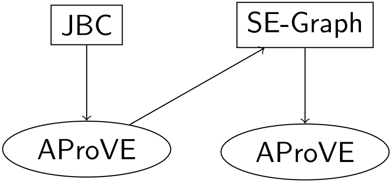
Overview



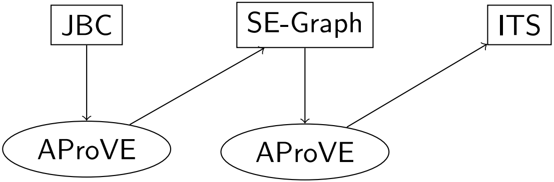
Overview



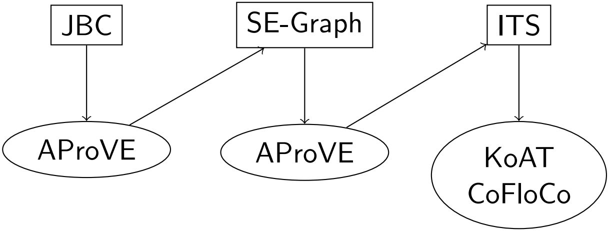
Overview



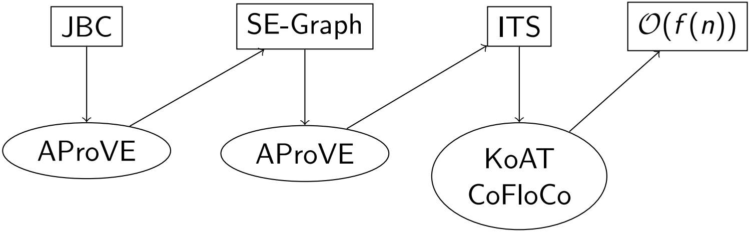
Overview



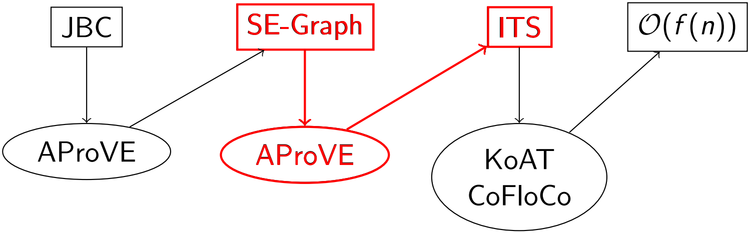
Overview



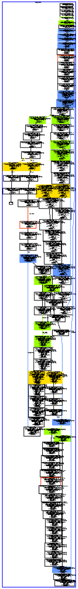
Overview



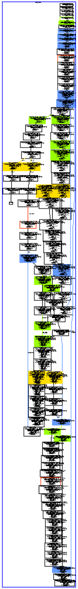
Overview



AProVE's Symbolic Evaluation Graphs

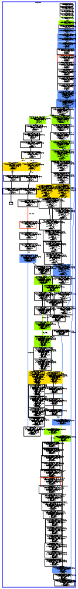


- AProVE: one of the most powerful termination and complexity analyzers



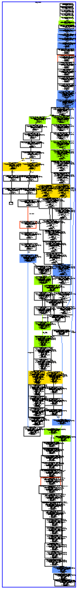
- AProVE: one of the most powerful termination and complexity analyzers
- SE-Graphs: developed for termination analysis

APROVE's Symbolic Evaluation Graphs



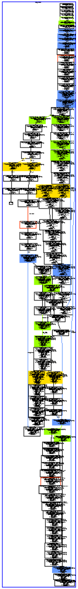
- AProVE: one of the most powerful termination and complexity analyzers
- SE-Graphs: developed for termination analysis
- intuition: CFG with invariants

AProVE's Symbolic Evaluation Graphs



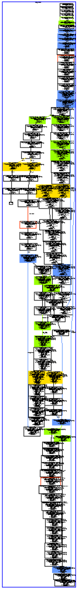
- AProVE: one of the most powerful termination and complexity analyzers
- SE-Graphs: developed for termination analysis
- intuition: CFG with invariants
  - node $\Longleftrightarrow$ program location

AProVE's Symbolic Evaluation Graphs



- AProVE: one of the most powerful termination and complexity analyzers
- SE-Graphs: developed for termination analysis
- intuition: CFG with invariants
  - node $\iff$ program location
  - node-content $\iff$ invariant

AProVE's Symbolic Evaluation Graphs



- AProVE: one of the most powerful termination and complexity analyzers
- SE-Graphs: developed for termination analysis
- intuition: CFG with invariants
  - node $\Longleftrightarrow$ program location
  - node-content $\Longleftrightarrow$ invariant
- details: see ...
  - Otto et al. RTA '10
  - Brockschmidt et al., RTA '11
  - Brockschmidt et al., FoVeOOS '11
  - Brockschmidt et al., CAV '12
  - ...

Nodes

$$\boxed{\begin{array}{l} \texttt{New List} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0 \end{array}}$$

Nodes

$$\boxed{\begin{array}{l}\texttt{New List} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0\end{array}}$$

Invariants:

- $\texttt{n} \geq 0$

Nodes

$$\begin{array}{|l|}\hline \texttt{New List} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0 \\ \hline \end{array}$$

Invariants:

- $\texttt{n} \geq 0$
- `this` is a tree

Nodes

$$\begin{array}{|l|}\hline \texttt{New List} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0 \\ \hline \end{array}$$

Invariants:

- $\texttt{n} \geq 0$
- `this` is a tree
  - otherwise: o_1!

Nodes

$\boxed{\begin{array}{l} \texttt{New List} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0 \end{array}}$

Invariants:

- $\texttt{n} \geq 0$
- `this` is a tree
    - otherwise: o_1!
- `this` and `r` don't share

Nodes

$$\boxed{\begin{array}{l}\texttt{New List} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0\end{array}}$$

Invariants:

- $\texttt{n} \geq 0$
- `this` is a tree
  - otherwise: $o_1!$
- `this` and `r` don't share
  - otherwise: $o_1 \bigvee o_2$

Goal: Transform SE-Graph to Integer Transition System

```
start(o522', i190) ->
  sort_ConstantStackPush_1(o522', i190)
sort_ConstantStackPush_1(o1) ->
  sort_Load_573(o1, 0, o1, o3', i1') |
  -o1 < i1' && o1 > 0 && o3' >= 0 && i1' < o1 && o3' < o1
sort_EQ_744(o529, x, i147, o531, o530, i172) ->
  sort_Inc_750(o529, i147, o531, o530, i172) |
  0 <= i147 && o530 >= 0 && o531 > 0 && o529 > 0 && x = 0
member_NE_734(i193, x, o521, o507, o509, o522, o508, i172) ->
  sort_EQ_744(o507, 1, i193, o509, o508, i172) |
  o509 > 0 && 0 <= i193 && o522 >= 0 && o508 >= 0 && o507 > 0 && o521 > 0 && x = i193
member_NE_734(i193, i147, o521, o507, o509, o522, o508, i172) ->
  member_Load_720(i147, o522, o507, o509, o508, i172) |
  o509 > 0 && 0 <= i147 && o522 >= 0 && o508 >= 0 && o521 > 0 && o507 > 0 && ...
sort_EQ_744(o529, x, i147, o531, o530, i172) ->
  sort_Inc_750(o529, i147, o542'1, o530, i172) |
  0 <= i147 && 0 <= 1 && o530 >= 0 && o542'1 > 0 && o531 > 0 && o529 > 0 && ...
max_Load_653(o438, i188, o439, i147, o441, o440, i172) ->
  max_NULL_654(o438, i188, o439, i147, o441, o440, i172) |
  o440 >= 0 && o441 > 0 && o439 > 0 && 0 <= i188 && o438 >= 0 && 0 <= i147
max_NULL_654(x, i188, o439, i147, o441, o440, i172) ->
  member_Load_720(i147, o439, o439, o441, o440, i172) |
  i188 >= i147 && 0 <= i147 && o440 >= 0 && o439 >= 0 && 0 <= i188 && o439 > 0 && ...
max_FieldAccess_679(o453, i188, o439, i147, o441, o454, i190, o440, i172) ->
  max_Load_653(o454, i188, o439, i147, o441, o440, i172) |
  o453 > 0 && 0 <= i147 && o439 > 0 && 0 <= i188 && o441 > 0 && o440 >= 0 && o454 >= 0
max_NULL_654(o449, i188, o439, i147, o441, o440, i172) ->
  max_LE_668(i190', i188, o449, o439, i147, o441, o454', o440, i172) |
  -o449 < i190' && 0 <= i147 && o440 >= 0 && o449 > 0 && o441 > 0 && 0 <= i188 && ...
...
```

Integer Transition System (ITSs)

rule-based representation of Integer Programs

Example

$$
\begin{array}{lcll}
f_{\mathsf{start}}(x) & \to & f(x) & \\
f(x) & \to & f(x-z) & \mid \quad x > 0 \wedge z > 0
\end{array}
$$

Integer Transition System (ITSs)

rule-based representation of Integer Programs

Example

$$\begin{array}{lcll} f_{\mathsf{start}}(x) & \to & f(x) & \\ f(x) & \to & f(x-z) & \mid \quad x > 0 \wedge z > 0 \end{array}$$

$f_{\mathsf{start}}(3)$

Integer Transition System (ITSs)

rule-based representation of Integer Programs

Example

$$\begin{array}{lcll} f_{\text{start}}(x) & \to & f(x) & \\ f(x) & \to & f(x-z) & \mid \quad x > 0 \land z > 0 \end{array}$$

$f_{\text{start}}(3) \to f(3)$

Integer Transition System (ITSs)

rule-based representation of Integer Programs

Example

$$\begin{array}{lcll} f_{\text{start}}(x) & \to & f(x) & \\ f(x) & \to & f(x-z) & \mid \quad x > 0 \land z > 0 \end{array}$$

$f_{\text{start}}(3) \to f(3) \to f(1)$

Integer Transition System (ITSs)

rule-based representation of Integer Programs

Example

$$\begin{array}{lcll} f_{\text{start}}(x) & \to & f(x) & \\ f(x) & \to & f(x-z) & \mid \quad x > 0 \land z > 0 \end{array}$$

$f_{\text{start}}(3) \to f(3) \to f(1) \to f(-2)$

SE-Graph $\rightarrow$ ITS

- translate each edge to a rule

SE-Graph $\rightarrow$ ITS

- translate each edge to a rule
- challenges

SE-Graph $\to$ ITS

- translate each edge to a rule
- challenges
  - abstract objects to integers

SE-Graph $\rightarrow$ ITS

- translate each edge to a rule
- challenges
  - abstract objects to integers
  - encode semantics of JBC instructions

SE-Graph $\rightarrow$ ITS

- translate each edge to a rule
- challenges
  - **abstract objects to integers**
  - encode semantics of JBC instructions

$$\boxed{\begin{array}{l} \texttt{New List} \mid \texttt{this}: o_1, \texttt{n}: i_1, \texttt{r}: o_2 \mid \varepsilon \\ o_1: \texttt{List},\ o_2: \texttt{List} \\ i_1 \geq 0 \end{array}} \quad \curvearrowright \quad f(o_1, i_1, o_2)$$

Size Abstraction

objects are graphs $\curvearrowright$ number of nodes

Size Abstraction

objects are graphs $\curvearrowright$ number of nodes

Example

$$5 \longrightarrow 3 \longrightarrow 7$$

Size Abstraction

objects are graphs $\curvearrowright$ number of nodes

Example


```
...
while (this.max() >= n){
  if (this.member(n))
    r = new List(n,r);
  n++;
}
...
```

Size Abstraction

objects are graphs $\curvearrowright$ number of nodes

Example

$5 \longrightarrow 3 \longrightarrow 7$

```
...
while (this.max() >= n){
  if (this.member(n))
    r = new List(n,r);
  n++;
}
...
```

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Size Abstraction

objects are graphs $\curvearrowright$ number of nodes

Example

$$\| 5 \longrightarrow 3 \longrightarrow 7 \| = 18$$

```
...
while (this.max() >= n){
  if (this.member(n))
    r = new List(n,r);
  n++;
}
...
```

$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$

Size Abstraction

objects are graphs $\curvearrowright$ number of nodes

Example

$$\|5 \longrightarrow 3 \longrightarrow 7\| = 18$$

```
...
while (this.max() >= n){
  if (this.member(n))
    r = new List(n,r);
  n++;
}
...
```

$$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$$

$$\curvearrowright \mathcal{O}(\|\texttt{this}\|^2)$$

SE-Graph $\rightarrow$ ITS

- translate each edge to a rule
- challenges
  - **abstract objects to integers**
  - encode semantics of JBC instructions

$$\boxed{\begin{array}{l} \texttt{New List} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0 \end{array}} \quad \curvearrowright \quad f(o_1, i_1, o_2)$$

SE-Graph $\rightarrow$ ITS

- translate each edge to a rule
- challenges
  - abstract objects to integers
  - **encode semantics of JBC instructions**

$$\boxed{\begin{array}{l} \texttt{New List} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0 \end{array}} \longrightarrow \boxed{\begin{array}{l} \ldots \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_3 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0 \end{array}} \quad \curvearrowright \quad f(o_1, i_1, o_2) \rightarrow g(o_1, i_1, o_2, o_3) \mid \varphi$$

Encoding New

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Encoding New

$$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$$

A Fresh List Instance

$0 \longrightarrow$ `null`

Encoding New

$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$

A Fresh List Instance

$\| (0) \longrightarrow \texttt{null} \| = 1$

Encoding New

$$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$$

A Fresh List Instance

$$\| \, 0 \longrightarrow \texttt{null} \, \| = 1$$

Example (Create New List Instance)

$$\begin{array}{l} \texttt{New List} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0 \end{array} \longrightarrow \begin{array}{l} \texttt{...} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid o_3 \\ o_1 : \texttt{List},\ o_2 : \texttt{List},\ o_3 : \texttt{List} \\ i_1 \geq 0 \end{array}$$

Encoding New

$$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$$

A Fresh List Instance

$$\|(0) \longrightarrow \texttt{null}\| = 1$$

Example (Create New List Instance)

$$\boxed{\begin{array}{l}\texttt{New List} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0\end{array}} \longrightarrow \boxed{\begin{array}{l}\ldots \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid o_3 \\ o_1 : \texttt{List},\ o_2 : \texttt{List},\ o_3 : \texttt{List} \\ i_1 \geq 0\end{array}}$$

$$f(\|o_1\|, i_1, \|o_2\|) \to g(\|o_1\|, i_1, \|o_2\|, \|o_3\|) \mid \|o_1\| \geq 0 \land \|o_2\| \geq 0 \land i_1 \geq 0 \land \|o_3\| = 1$$

Encoding New

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

A Fresh List Instance

$\| 0 \longrightarrow \texttt{null} \| = 1$

Example (Create New List Instance)

$$\begin{array}{l} \texttt{New List} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0 \end{array} \longrightarrow \begin{array}{l} \texttt{...} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid o_3 \\ o_1 : \texttt{List},\ o_2 : \texttt{List},\ o_3 : \texttt{List} \\ i_1 \geq 0 \end{array}$$

$$f(\ o_1\ , i_1,\ o_2\ ) \to g(\ o_1\ , i_1,\ o_2\ ,\ o_3\ ) \mid\ o_1\ \geq 0 \wedge\ o_2\ \geq 0 \wedge i_1 \geq 0 \wedge\ o_3\ = 1$$

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Encoding Write Accesses

$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$

Write to `value`

$\| 5 \longrightarrow 3 \longrightarrow 7 \| = 18$

Encoding Write Accesses

$$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$$

Write to `value`

$\|5 \rightarrow 3 \rightarrow 7\| = 18 \curvearrowright -9 \rightarrow 3 \rightarrow 7$

Encoding Write Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Write to `value`

$\|5 \longrightarrow 3 \longrightarrow 7\| = 18 \curvearrowright \|{-9} \longrightarrow 3 \longrightarrow 7\| = 22$

Encoding Write Accesses

$$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$$

Write to `value`

$$\|5 \longrightarrow 3 \longrightarrow 7\| = 18 \curvearrowright \|\text{-9} \longrightarrow 3 \longrightarrow 7\| = 22 \leq 18 + |-9|$$

Encoding Write Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Write to value

$\| (5) \rightarrow (3) \rightarrow (7) \| = 18 \curvearrowright \| (-9) \rightarrow (3) \rightarrow (7) \| = 22 \leq 18 + |-9|$

Example (Write i_1 to o_1.value)

Write value $|$ this : o_1, n : i_1, r : o_2 $|$ o_1, i_1
o_1 : List, o_2 : List
$i_1 \geq 0$

$\longrightarrow$

... $|$ this : o_1, n : i_1, r : o_2 $|$ ε
o_1 : List, o_2 : List
$i_1 \geq 0$

Encoding Write Accesses

$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$

Write to `value`

$\|5 \rightarrow 3 \rightarrow 7\| = 18 \;\curvearrowright\; \|{-9} \rightarrow 3 \rightarrow 7\| = 22 \leq 18 + |-9|$

Example (Write i_1 to o_1.`value`)

`Write value` | `this` : o_1, `n` : i_1, `r` : o_2 | o_1, i_1
o_1 : `List`, o_2 : `List`
$i_1 \geq 0$

$\longrightarrow$

. . . | `this` : o_1, `n` : i_1, `r` : o_2 | ε
o_1 : `List`, o_2 : `List`
$i_1 \geq 0$

$$f(o_1, i_1, o_2) \;\rightarrow\; g(o_1', i_1, o_2) \;\mid\; \ldots \wedge i_1 \geq 0 \wedge o_1 + i_1 \geq o_1'$$

Encoding Write Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Write to `value`

$$\|(5) \longrightarrow (3) \longrightarrow (7)\| = 18 \ \curvearrowright\ \|(-9) \longrightarrow (3) \longrightarrow (7)\| = 22 \leq 18 + |-9|$$

Example (Write i_1 to o_1.`value`)

$$\boxed{\begin{array}{l} \texttt{Write value} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid o_1, i_1 \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ \cancel{i_1 \geq 0} \end{array}} \longrightarrow \boxed{\begin{array}{l} \ldots \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ \cancel{i_1 \geq 0} \end{array}}$$

$$f(o_1, i_1, o_2) \ \rightarrow\ g(o_1', i_1, o_2) \ \mid\ \ldots \wedge i_1 \geq 0 \wedge o_1 + i_1 \geq o_1'$$

Encoding Write Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Write to `value`

$$\|5 \longrightarrow 3 \longrightarrow 7\| = 18 \;\curvearrowright\; \|{-9} \longrightarrow 3 \longrightarrow 7\| = 22 \leq 18 + |-9|$$

Example (Write i_1 to o_1.`value`)

`Write value` | `this` : o_1, `n` : i_1, `r` : o_2 | o_1, i_1
o_1 : `List`, o_2 : `List`
$\cancel{i_1 \geq 0}$

$\longrightarrow$

`...` | `this` : o_1, `n` : i_1, `r` : o_2 | ε
o_1 : `List`, o_2 : `List`
$\cancel{i_1 \geq 0}$

$$
\begin{array}{lclcl}
f(o_1, i_1, o_2) & \rightarrow & g(o_1', i_1, o_2) & \mid & \ldots \wedge i_1 \geq 0 \wedge o_1 + i_1 \geq o_1' \\
f(o_1, i_1, o_2) & \rightarrow & g(o_1', i_1, o_2) & \mid & \ldots \wedge i_1 < 0 \wedge o_1 - i_1 \geq o_1'
\end{array}
$$

Encoding Write Accesses

$$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$$

Write to value

$$\|(5) \longrightarrow (3) \longrightarrow (7)\| = 18 \;\curvearrowright\; \|(\text{-}9) \longrightarrow (3) \longrightarrow (7)\| = 22 \leq 18 + |-9|$$

Example (Write i_1 to o_1.value)

$\texttt{Write value} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid o_1, i_1$
$o_1 : \texttt{List},\ o_2 : \texttt{List}$
$\cancel{i_1 \geq 0}$, $o_1 \searrow\!\!\swarrow o_2$

$\longrightarrow$

$\ldots \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon$
$o_1 : \texttt{List},\ o_2 : \texttt{List}$
$\cancel{i_1 \geq 0}$, $o_1 \searrow\!\!\swarrow o_2$

$$\begin{array}{lclcl} f(o_1, i_1, o_2) & \to & g(o_1', i_1, o_2) & \mid & \ldots \wedge i_1 \geq 0 \wedge o_1 + i_1 \geq o_1' \\ f(o_1, i_1, o_2) & \to & g(o_1', i_1, o_2) & \mid & \ldots \wedge i_1 < 0 \wedge o_1 - i_1 \geq o_1' \end{array}$$

Encoding Write Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Write to value

$\|\,5 \longrightarrow 3 \longrightarrow 7\,\| = 18 \;\curvearrowright\; \|\,\text{-9} \longrightarrow 3 \longrightarrow 7\,\| = 22 \leq 18 + |-9|$

Example (Write i_1 to o_1.value)

$$\begin{array}{l} \texttt{Write value} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid o_1, i_1 \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ \cancel{i_1 \geq 0},\ o_1 \curlyvee o_2 \end{array} \longrightarrow \begin{array}{l} \ldots \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid \varepsilon \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ \cancel{i_1 \geq 0},\ o_1 \curlyvee o_2 \end{array}$$

$$\begin{array}{lcl} f(o_1, i_1, o_2) & \to & g(o_1', i_1, o_2') \mid \ldots \wedge i_1 \geq 0 \wedge o_1 + i_1 \geq o_1' \wedge o_2 + i_1 \geq o_2' \\ f(o_1, i_1, o_2) & \to & g(o_1', i_1, o_2') \mid \ldots \wedge i_1 < 0 \wedge o_1 - i_1 \geq o_1' \wedge o_2 - i_1 \geq o_2' \end{array}$$

Encoding Read Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Encoding Read Accesses

$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$

Read From `next`

$\| (5) \longrightarrow (3) \longrightarrow (7) \| = 18$

Encoding Read Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Read From `next`

$\| 5 \longrightarrow 3 \longrightarrow 7 \| = 18 \curvearrowright 3 \longrightarrow 7$

Encoding Read Accesses

$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$

Read From `next`

$$\| 5 \longrightarrow 3 \longrightarrow 7 \| = 18 \;\curvearrowright\; \| 3 \longrightarrow 7 \| = 12$$

Encoding Read Accesses

$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$

Read From `next`

$\left\| (5) \longrightarrow (3) \longrightarrow (7) \right\| = 18 \curvearrowright \left\| (3) \longrightarrow (7) \right\| = 12 < 18$

Encoding Read Accesses

$$\|o\| = \#\text{reachable objects} + \sum \text{absolute values of reachable integers}$$

Read From `next`

$$\|\,(5) \longrightarrow (3) \longrightarrow (7)\,\| = 18 \;\curvearrowright\; \|\,(3) \longrightarrow (7)\,\| = 12 < 18$$

Example (Read o_1.next)

$$\boxed{\begin{array}{l} \texttt{Read next} \mid \texttt{this}: o_1, \texttt{n}: i_1, \texttt{r}: o_2 \mid o_1 \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0 \end{array}} \longrightarrow \boxed{\begin{array}{l} \ldots \mid \texttt{this}: o_1, \texttt{n}: i_1, \texttt{r}: o_2 \mid o_3 \\ o_1 : \texttt{List},\ o_2 : \texttt{List},\ o_3 : \texttt{List} \\ i_1 \geq 0,\ o_1 \searrow\!\!\swarrow\ o_3 \end{array}}$$

Encoding Read Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Read From `next`

$\| 5 \longrightarrow 3 \longrightarrow 7 \| = 18 \;\curvearrowright\; \| 3 \longrightarrow 7 \| = 12 < 18$

Example (Read o_1.`next`)

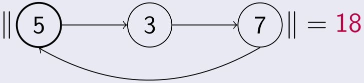
$$\boxed{\begin{array}{l} \texttt{Read next} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid o_1 \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0 \end{array}} \longrightarrow \boxed{\begin{array}{l} \ldots \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid o_3 \\ o_1 : \texttt{List},\ o_2 : \texttt{List},\ o_3 : \texttt{List} \\ i_1 \geq 0,\ o_1 \bigvee o_3 \end{array}}$$

$$f(o_1, i_1, o_2) \rightarrow g(o_1, i_1, o_2, o_3) \mid \ldots \wedge o_1 > o_3$$

Encoding Read Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Read From `next`



Example (Read o_1.next)

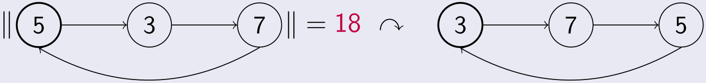
$$\boxed{\begin{array}{l}\texttt{Read next} \mid \texttt{this}: o_1, \texttt{n}: i_1, \texttt{r}: o_2 \mid o_1 \\ o_1 : \texttt{List},\ o_2 : \texttt{List} \\ i_1 \geq 0\end{array}} \longrightarrow \boxed{\begin{array}{l}\ldots \mid \texttt{this}: o_1, \texttt{n}: i_1, \texttt{r}: o_2 \mid o_3 \\ o_1 : \texttt{List},\ o_2 : \texttt{List},\ o_3 : \texttt{List} \\ i_1 \geq 0,\ o_1 \searrow\!\swarrow o_3\end{array}}$$

$$f(o_1, i_1, o_2) \to g(o_1, i_1, o_2, o_3) \mid \ldots \wedge o_1 > o_3$$

Encoding Read Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Read From `next`



Example (Read o_1.`next`)

`Read next` | `this` : o_1, `n` : i_1, `r` : o_2 | o_1
o_1 : `List`, o_2 : `List`
$i_1 \geq 0$

$\longrightarrow$

. . . | `this` : o_1, `n` : i_1, `r` : o_2 | o_3
o_1 : `List`, o_2 : `List`, o_3 : `List`
$i_1 \geq 0$, $o_1 \searrow\!\swarrow o_3$

$$f(o_1, i_1, o_2) \rightarrow g(o_1, i_1, o_2, o_3) \mid \ldots \wedge o_1 > o_3$$

Encoding Read Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Read From `next`

Example (Read o_1.`next`)

`Read next` | `this` : o_1, `n` : i_1, `r` : o_2 | o_1
o_1 : `List`, o_2 : `List`
$i_1 \geq 0$

$\longrightarrow$

... | `this` : o_1, `n` : i_1, `r` : o_2 | o_3
o_1 : `List`, o_2 : `List`, o_3 : `List`
$i_1 \geq 0$, $o_1 \searrow\!\swarrow o_3$

$$f(o_1, i_1, o_2) \to g(o_1, i_1, o_2, o_3) \mid \ldots \wedge o_1 > o_3$$

Encoding Read Accesses

$\|o\|$ = #reachable objects + $\sum$ absolute values of reachable integers

Read From `next`

Example (Read o_1.`next`)

$\texttt{Read next} \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid o_1$
$o_1 : \texttt{List},\ o_2 : \texttt{List}$
$i_1 \geq 0,\ o_1!$

$\longrightarrow$

$\ldots \mid \texttt{this} : o_1, \texttt{n} : i_1, \texttt{r} : o_2 \mid o_3$
$o_1 : \texttt{List},\ o_2 : \texttt{List},\ o_3 : \texttt{List}$
$i_1 \geq 0,\ o_1 \not\curlyvee o_3,\ o_1!,\ o_3!$

$$f(o_1, i_1, o_2) \to g(o_1, i_1, o_2, o_3) \mid \ldots \wedge o_1 > o_3$$

Encoding Read Accesses

$\|o\| = \#$reachable objects $+ \sum$ absolute values of reachable integers

Read From `next`

Example (Read o_1.`next`)

`Read next` | `this` : o_1, `n` : i_1, `r` : o_2 | o_1
o_1 : `List`, o_2 : `List`
$i_1 \geq 0$, $o_1!$

$\longrightarrow$

... | `this` : o_1, `n` : i_1, `r` : o_2 | o_3
o_1 : `List`, o_2 : `List`, o_3 : `List`
$i_1 \geq 0$, $o_1 \mathrel{\vee\!\!\!/} o_3$, $o_1!$, $o_3!$

$$f(o_1, i_1, o_2) \rightarrow g(o_1, i_1, o_2, o_3) \mid \ldots \wedge o_1 \geq o_3$$

Conclusion And Experiments

Done

Done

- lifted AProVE's termination technique to complexity

Conclusion And Experiments

Done

- lifted AProVE's termination technique to complexity
- used to check programs for DoS vulnerabilities

Conclusion And Experiments

Done

- lifted AProVE's termination technique to complexity
- used to check programs for DoS vulnerabilities

Future Work

Conclusion And Experiments

Done

- lifted AProVE's termination technique to complexity
- used to check programs for DoS vulnerabilities

Future Work

- language features: Recursion, Java 8

Conclusion And Experiments

Done

- lifted AProVE's termination technique to complexity
- used to check programs for DoS vulnerabilities

Future Work

- language features: Recursion, Java 8
- more expressive heap predicates (see PhD-iFM proceedings)

Conclusion And Experiments

Done

- lifted AProVE's termination technique to complexity
- used to check programs for DoS vulnerabilities

Future Work

- language features: Recursion, Java 8
- more expressive heap predicates (see PhD-iFM proceedings)

Experiments on 211 programs from the TPDB

	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	$\mathcal{O}(n \cdot \log n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^{>3})$	?	Success
AProVE	28	0	102	0	13	2	4	62	71 %
COSTA	10	4	45	3	5	0	1	143	32 %

Thank You!

Demo!

Beyond Time Complexity

- attach <u>costs</u> to rules

Beyond Time Complexity

- attach <u>costs</u> to rules
- model network traffic, loop iterations, heap space, ...

Beyond Time Complexity

- attach costs to rules
- model network traffic, loop iterations, heap space, ...

Example

`new`: cost $= 1$

Beyond Time Complexity

- attach costs to rules
- model network traffic, loop iterations, heap space, ...

Example

`new`: cost $=$ 1

`anewarray`: cost $=$ size of the new array

Beyond Time Complexity

- attach costs to rules
- model network traffic, loop iterations, heap space, ...

Example

`new`: cost $=1$

`anewarray`: cost $=$ size of the new array

all other instructions: cost $=0$

Beyond Time Complexity

- attach costs to rules
- model network traffic, loop iterations, heap space, ...

Example

`new`: cost $= 1$

`anewarray`: cost $=$ size of the new array

all other instructions: cost $= 0$

$\curvearrowright$ models auxiliary heap space