Lightweight Verification of Array Indexing

Martin Kellogg*, Vlastimil Dort**, Suzanne Millstein*, Michael D. Ernst*

* University of Washington, Seattle
** Charles University, Prague
The problem: unsafe array indexing

- In unsafe languages (C): buffer overflow!
- In managed languages (Java, C#, etc.): exception, program crashes
The state of the art

Strength of guarantees

Practical for developers
The state of the art

Strength of guarantees

Practical for developers

Coq

KeY

Clousot
The state of the art

Strength of guarantees

Practical for developers

Coq  KeY  Clousot  Coverity  FindBugs
The state of the art

- Coq
- KeY
- Clousot
- The Index Checker (this talk)

Strength of guarantees

Practical for developers:
- FindBugs
- Coverity
Problems with complex analyses

- false positives

- annotation burden

- complex analyses are hard to predict
Problems with complex analyses

- false positives
  - bounds checking is hard → complex analysis
  - complex analysis → harder to implement
  - harder to implement → more false positives

- annotation burden

- complex analyses are hard to predict
Problems with complex analyses

- false positives
  ● bounds checking is hard → complex analysis
  ● complex analysis → harder to implement
  ● harder to implement → more false positives

- annotation burden
  ● complex analysis → complex annotations

- complex analyses are hard to predict
Problems with complex analyses

- false positives
  - bounds checking is hard $\rightarrow$ complex analysis
  - complex analysis $\rightarrow$ harder to implement
  - harder to implement $\rightarrow$ more false positives

- annotation burden
  - complex analysis $\rightarrow$ complex annotations

- complex analyses are hard to predict
Insight:

Fundamental problem is complex analyses!
Cooperating simple analyses

Solve all three problems:
Cooperating simple analyses

Solve all three problems:

- simpler implementation $\rightarrow$ fewer false positives
Cooperating simple analyses

Solve all three problems:

● simpler implementation $\rightarrow$ fewer false positives
● simpler abstractions $\rightarrow$ easier to write annotations
Cooperating simple analyses

Solve all three problems:

● simpler implementation $\rightarrow$ fewer false positives
● simpler abstractions $\rightarrow$ easier to write annotations
● simpler analysis $\rightarrow$ simpler to predict
Proving an array access safe

T[] a = ...;
int i = ...;
... a[i] ...

We need to show that:
• i is an index for a
Proving an array access safe

```java
T[] a = ...;
int i = ...;
... a[i] ...
```

We need to show that:

- $i$ is an index for $a$
- $i \geq 0$
- $i < a.length$
Proving an array access safe

\[
T[\ ] \ a = \ ...; \\
\textbf{int} \ i = \ ...; \\
... \ a[i] \ ... \\
\]

We need to show that:

- \(i\) is an index for \(a\)
- \(i \geq 0\)  \hspace{1cm} \text{A lower bound on } i
- \(i < a.\text{length}\)  \hspace{1cm} \text{An upper bound on } i
A type system for lower bounds

\[
\begin{align*}
T & \quad \uparrow \quad \@\text{LowerBoundUnknown} \ \text{int} \ i \\
\uparrow & \quad \uparrow \\
i \geq -1 & \quad \@\text{GTENegativeOne} \ \text{int} \ i \\
\uparrow & \quad \uparrow \\
i \geq 0 & \quad \@\text{NonNegative} \ \text{int} \ i \\
\uparrow & \quad \uparrow \\
i \geq 1 & \quad \@\text{Positive} \ \text{int} \ i
\end{align*}
\]
A type system for lower bounds

\[
\begin{align*}
T &\quad \text{@LowerBoundUnknown int } i \\
\uparrow &\quad \uparrow \\
\text{i} &\geq -1 \quad \text{@GTENegativeOne int } i \\
\uparrow &\quad \uparrow \\
\text{i} &\geq 0 \quad \text{@NonNegative int } i \\
\uparrow &\quad \uparrow \\
\text{i} &\geq 1 \quad \text{@Positive int } i
\end{align*}
\]
A type system for upper bounds

```java
if (i >= 0 && i < a.length) {
    a[i] = ...
}
```
A type system for upper bounds

```java
if (i >= 0 && i < a.length) {
    a[i] = ... 
}
```

```java
i < a.length @LTLLengthOf("a") int i
```
Type systems

Linear inequalities
\( i < j \)

Negative indices
\(| i | < a.length\)

Equal lengths
\( a.length = b.length\)

Minimum lengths
\( a.length > 10 \)

Lower bounds
\( i \geq 0 \)

Upper bounds
\( i < a.length \)
Type systems

- Linear inequalities: \( i < j \)
- Negative indices: \( |i| < a.length \)
- Equal lengths: \( a.length = b.length \)
- Minimum lengths: \( a.length > 10 \)
- Lower bounds: \( i \geq 0 \)
- Upper bounds: \( i < a.length \)
A type system for minimum array lengths

```java
if (a.length >= 3) {
    a[2] = ...;
}
```
A type system for minimum array lengths

```java
if (a.length >= 3) {
    a[2] = ...;
}
```

\[ a.length \geq i \quad \text{T \, @MinLen(i) \, [] \, a} \]
Evaluation

Three case studies:
- Google Guava (two packages)
- JFreeChart
- plume-lib

Comparison to existing tools:
- FindBugs, KeY, Clousot
# Case Studies

<table>
<thead>
<tr>
<th></th>
<th>Guava</th>
<th>JFreeChart</th>
<th>plume-lib</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of code</td>
<td>10,694</td>
<td>94,233</td>
<td>14,586</td>
<td>119,503</td>
</tr>
<tr>
<td>Bugs found</td>
<td>5</td>
<td>64</td>
<td>20</td>
<td>89</td>
</tr>
<tr>
<td>Annotations</td>
<td>510</td>
<td>2,938</td>
<td>241</td>
<td>3,689</td>
</tr>
<tr>
<td>False positives</td>
<td>138</td>
<td>386</td>
<td>43</td>
<td>567</td>
</tr>
<tr>
<td>Java casts</td>
<td>222</td>
<td>2,740</td>
<td>219</td>
<td>3,181</td>
</tr>
</tbody>
</table>
## Comparison to other tools: confirmed bugs

<table>
<thead>
<tr>
<th>Approach</th>
<th>Types</th>
<th>Bug finder</th>
<th>Verif. w/ solver</th>
<th>Abs. interpret.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>Index Checker</td>
<td>FindBugs</td>
<td>KeY</td>
<td>Clousot</td>
</tr>
<tr>
<td>True Positives</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False Negatives</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time (100k LoC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparison to other tools: confirmed bugs

<table>
<thead>
<tr>
<th>Approach</th>
<th>Types</th>
<th>Bug finder</th>
<th>Verif. w/ solver</th>
<th>Abs. interpret.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>Index Checker</td>
<td>FindBugs</td>
<td>KeY</td>
<td>Clousot</td>
</tr>
<tr>
<td>True Positives</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False Negatives</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time (100k LoC)</td>
<td>~10 minutes</td>
<td>~1 minute</td>
<td>cannot scale</td>
<td>~200 minutes</td>
</tr>
</tbody>
</table>
Comparison to other tools: confirmed bugs

<table>
<thead>
<tr>
<th>Approach</th>
<th>Types</th>
<th>Bug finder</th>
<th>Verif. w/ solver</th>
<th>Abs. interpret.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tool</strong></td>
<td><strong>Index Checker</strong></td>
<td><strong>FindBugs</strong></td>
<td><strong>KeY</strong></td>
<td><strong>Clousot</strong></td>
</tr>
<tr>
<td>True Positives</td>
<td>18/18</td>
<td>0/18</td>
<td>9/18</td>
<td>16/18</td>
</tr>
<tr>
<td>False Negatives</td>
<td>0/18</td>
<td>18/18</td>
<td>1/18</td>
<td>2/18</td>
</tr>
<tr>
<td>Time (100k LoC)</td>
<td>~10 minutes</td>
<td>~1 minute</td>
<td>cannot scale</td>
<td>~200 minutes</td>
</tr>
</tbody>
</table>
Using the Index Checker

- Distributed with Checker Framework

www.checkerframework.org
Contributions

- A methodology: simple, cooperative type systems
- An analysis: abstractions for array indexing
- An implementation and evaluation for Java
- Verifying the absence of array bounds errors in real codebases (and finding bugs in the process!)