Finding Inter-procedural Bugs at Scale with Infer

Jules Villard <jul@fb.com>
Facebook London
Infer

- Open-source static analyser
- Inter-procedural analyses + linters
- For Java and C/C++/Objective-C
Two Frontends: clang and Java
And quite a few build system integrations

- **PROJECT**
  - SOURCE CODE
  - BUILD SYSTEM

- **SOURCE FILES**
  - + COMPILER COMMANDS

- **SOURCE CODE**
  - + clang plugin

- **BUILD SYSTEM**
  - SOURCE FILES

- **CLANG AST**
  - + javac

- **JAVA BYTECODE**
  - to backend...

- **CLANG AST**
  - C/C++/ObjC frontend

- **REPORT**
  - linters

- **SIL**
  - Java frontend

- **SIL**
  - Java frontend

- **SIL**
  - C/C++/ObjC frontend

- **SIL**
  - linters
Infer architecture
Compositional, On-Demand Backend Architecture

"Allocates Memory" checker case study

```java
void foo() {
  ...
  Bar.bar();
  ...
}

@NoAllocation
void goo() {
  ...
  foo();
  ...
}
```

```java
void bar() {
  ...
  new MyObject();
  ...
}

void baz() {
  ...
}
```

Foo.java (SIL)

Bar.java (SIL)
Compositional, On-Demand Backend Architecture

"Allocates Memory" checker case study

```java
void foo() {
    Bar.bar();
}
```

```java
@NoAllocation
void goo() {
    foo();
}
```

---

```java
void bar() {
    new MyObject();
}
```

```java
void baz() {
    ...
}
```

---

**Foo.java** (SIL)

**Bar.java** (SIL)
### Interprocedural Analysis Case Study

#### Percentages of inter-procedural reports for different types of bugs

<table>
<thead>
<tr>
<th>Bug Type</th>
<th>One procedure One file</th>
<th>Interprocedural One file</th>
<th>Interprocedural Inter-file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocates Memory</td>
<td>0</td>
<td>2</td>
<td>98</td>
</tr>
<tr>
<td>Null Dereference (Java)</td>
<td>43</td>
<td>9</td>
<td>48</td>
</tr>
<tr>
<td>Null Dereference (Objective-C)</td>
<td>73</td>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td>RacerD</td>
<td>36</td>
<td>12</td>
<td>53</td>
</tr>
<tr>
<td>Bad Pointer Comparison (linter)</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Diff comments fit into usual workflow

Only report when:
- Warning is introduced by diff
- Warning is in file changed by diff
Analysing a Diff

"Allocates Memory" checker case study

```java
--- Foo.java
+++ Foo.java
@NoAllocation
void goo() {
  ...
  +  foo();
  ...
}
```
Analysing a Diff

"Allocates Memory" checker case study

--- Foo.java
+++ Foo.java
@NoAllocation
void goo() {
    ...
    Bar.bar();
    ...
}  

--- Foo.java
+++ Foo.java
@NoAllocation
void goo() {
    ...
    @NoAllocation
    void goo() {
        ...
        foo();
    }
}  

--- Foo.java
+++ Foo.java
@NoAllocation
void goo() {
    ...
    foo();
    ...
}  

with diff

--- Bar.java
+++ Bar.java
void bar() {
    ...
    new MyObject();
    ...
}

@NoAllocation
void baz() {
    ...
}
Analysing a Diff

"Allocates Memory" checker case study

--- Foo.java
+++ Foo.java
@NoAllocation
void goo() {
  ...
+  foo();
  ...
}

Allocation via call to foo() line 10

@NoAllocation
void goo() {
  ...
  foo();
  ...
}

Allocation via call to bar() line 3

Allocation line 3

void bar() {
  ...
  new MyObject();
  ...
}

void baz() {
  ...
}

ERROR
Analysing a Diff

"Allocates Memory" checker case study

--- Foo.java
+++ Foo.java
@NoAllocation
void goo() {
  ...
  foo();
  ...
}

1: void foo() {
2:   ...
3:   Bar.bar();
4:   ...
5: }
6: @NoAllocation
7: void goo() {
8:   ...
9:   ...
10: }
11: }
12: }

1: void bar() {
2:   ...
3: }
4: }
5: }
6: }
7: void baz() {
8:   ...
9: }
10: }
11: }
12: }

Allocation line 3
Analysing a Diff

"Allocates Memory" checker case study

```java
--- Foo.java
+++ Foo.java
@NoAllocation
void goo() {
    ...
+    foo();
    ...
}
```

```
Bar.java
```
Analysing a Diff

"Allocates Memory" checker case study

--- Foo.java
+++ Foo.java
@NoAllocation
void goo() {
  ...
  + foo();
  ...
}

---

Diff - base =
ERROR foo() allocates memory on line 10

DIFFERENTIAL REPORT
Diff-Based Deployment

- Help developers move fast
- Easy to deploy new checks
Current status

- Infer runs on all Android + iOS diffs for Facebook, Messenger, Instagram, and WhatsApp
- 10ks of diffs analyzed per month
- 1ks of issues fixed per month (~70% fix rate)

Action taken is ground truth for success
Finding Inter-procedural Bugs at Scale with Infer

Jules Villard <jul@fb.com>
Facebook London