Homework 3
Posted Friday February 26, Due Tuesday March 12
50 points

Now that you are (more) familiar with Soot and the Class analysis framework, you will build a more complex class analysis, XTA. Add a new package `analysis.XTA` and add your XTA implementation in `public class XTAAnalysis extends Analysis` in this package. You may add analogous drivers to the ones in RTA to test locally. Follow directory structure as Submitty pulls your `analysis/XTA/XTAAnalysis.java` to test.

Below is a rehash of the XTA constraints we discussed in class.

1. Allocation:
   
   \[
   \begin{align*}
   \text{for each } & \text{ new } \ A \text{ in } m \text{ s.t. } m \in \text{ReachableMethods} \do \\
   & \{A\} \subseteq S_m \\
   \end{align*}
   \]

2. Virtual call:
   
   \[
   \begin{align*}
   \text{for each } & \text{ } x = y.n(z) \text{ in } m \text{ s.t. } m \in \text{ReachableMethods} \do \\
   & \text{for each } \ C \text{ in } S_m \cap \text{SubTypes(StaticType(y))} \do \\
   & \quad n'(\text{this}, p, \text{ret}) = \text{resolve}(C, n) \text{ // adds target to ReachableMethods} \\
   & \quad \{n'\} \subseteq \text{ReachableMethods} \\
   & \quad \{C\} \subseteq S_{n'} \text{ // adds receiver class to } S_{n'} \\
   & \quad S_m \cap \text{SubTypes(StaticType(p))} \subseteq S_{n'} \text{ // adds to } S_{n'} \text{ due to arguments} \\
   & \quad S_{n'} \cap \text{SubTypes(StaticType(ret))} \subseteq S_m \text{ // adds to } S_m \text{ from } S_{n'} \text{ due to return} \\
   \end{align*}
   \]

3. Field Read:

   \[
   \begin{align*}
   \text{for each } & \text{ } x = y.f \text{ in } m \text{ s.t. } m \in \text{ReachableMethods} \do \\
   & S_f \subseteq S_m \\
   \end{align*}
   \]

4. Field Write:

   \[
   \begin{align*}
   \text{for each } & \text{ } x.f = y \text{ in } m \text{ s.t. } m \in \text{ReachableMethods} \do \\
   & S_m \cap \text{SubTypes(StaticType(f))} \subseteq S_f \\
   \end{align*}
   \]

In addition, you must handle direct calls, static fields and array reads/writes.

Direct calls to static methods are straightforward. Direct calls to instance methods (i.e., these are methods such as constructors that have a receiver `this`) use the following constraint to pass the type of the receiver to the callee `n`:

\[
S_m \cap \text{SubTypes(StaticType(this))} \subseteq S_n
\]
Static field reads, \texttt{local = static\_field}, and writes, \texttt{static\_field = local}, are abstracted as \texttt{assignStmt} in the class analysis framework, where respectively, the right-hand-side or left-hand-side node is of kind \texttt{STATIC\_FIELD}.

Make sure you handle arrays. Ignoring arrays renders the XTA analysis unsound. Consider

```java
void m(X[] a) {
    X x = a[0];
    x.n();
}
```

and note that in general, the array argument may have been written \textit{anywhere} in the program.

Finally, as with RTA, display your result in \texttt{showResult}. Specifically, display all reachable methods \textit{m} in alphabetical order with all classes in \textit{S_m} in alphabetical order. For example, the expected output for \texttt{p2} is the following

Reachable methods:

```java
<A: int add(A)>
    === A
    === B

<A: void <init>()>
    === A
    === B

<A: void m()>  
    === A
    === B

<A: void main(java.lang.String[])>  
<A: void sm()>  
    === A
    === B

<B: void <init>()>
    === B
```

When you are done, push into your \texttt{hw02} repository and click Grade my Repository in Submitty. Submitty pulls your \texttt{analysis/XTA/XTAAnalysis.java} to test. Make sure you commit/push the entire \texttt{XTA} directory.