15-819O Program Analysis

Spring 2013

Jonathan Aldrich

Topic: Synchronization Removal Analysis, Object-Oriented Invariants

Readings:

Erik Ruf. Effective Synchronization Removal for Java. Programming Language Design and Implementation, 2000.

Peter Müller, Arnd Poetzsch-Heffter, and Gary T. Leavens. Modular Invariants for Layered Object Structures. Science of Computer Programming 62(3):253-286, 2006.

Reading Question Set 2 [due March 20th, 2013 at 11:59pm]

Read the papers cited above. Answer the 4 questions below in an email to the instructor (aldrich@cs.cmu.edu) with subject "RQ 2":

1. Relate Ruf's analysis to Steensgaard's alias analysis and to OO call graph construction using 1-CFA. In 2-3 sentences, describe what are the similarities and the differences in the techniques used.

2. Ruf uses summary-based, context-sensitive interprocedural analysis, but also uses annotations. In about a paragraph, explain his design, and discuss why it is different from the way that interprocedural analysis and annotations are used in the Hackett et al. buffer overrun analysis

3. Ruf's analysis combines aliasing and synchronization information in one lattice structure, rather than computing aliasing information in one analysis and using the results in a second analysis. In about a paragraph, discuss the benefits and drawbacks of this choice.

4. For each of the 4 examples below, write invariants for each class that enforce the required property. For each example, state what is the simplest technique from the Müller et al. paper (from simple to complex: classic, ownership, ownership/readonly, visibility) that can be used to verify it. If verifying the invariants requires an ownership technique, state which fields are rep, which are peer, and which are readonly.

   (The examples are on the next 2 pages. Note that it is illegal to assign to a field marked **final** *in Java. Note also that there is one* ***discussion question*** *after the examples.)*

EXAMPLE A

```
// invariant: elements of the list should hold increasing values
class ListCell {
        int val;
        ListCell next;
        // requires v < n.val
        ListCell(int v, ListCell n) { val = v; next = n; }
        // requires v >  n.val
        void insert(int v) {
                ListCell beforeCell = this;
                while (beforeCell.next != null && v > beforeCell.next.val)
                        beforeCell = beforeCell.next;
                beforeCell.next = new ListCell(v, beforeCell.next);
        }
}
```

EXAMPLE B

```
// invariant: the range should not be empty
class Range {
        int low;
        int high;
        Range(int lo, int hi) { low = lo; high = max(lo, hi); }
        void moveRange(int amt) { low += amt; high += amt; }
}
```

EXAMPLE C

```
// invariant: neighboring cells in a list should link consistently to each other
class DoubleListCell {
        int val;
        DoubleListCell next;
        DoubleListCell prev;
        DoubleListCell(int v, DoubleListCell  afterCell) {
                val = v;
                if (afterCell == null) {
                        next = this;
                        prev = this;
                } else {
                        next = afterCell.next;
                        prev =afterCell;
                        afterCell.next = this;
                        next.prev = this;
}       }       }
```

EXAMPLE D

```
// invariant: the size of the list is accurate
class SizedCell {
        int val;
        final int size;
        final SizedCell next;
        SizedCell(int v, SizedCell n) {
                val = v;
                next = n;
                size = (n==null) ? 1 : (n.size+1);
        }
}
```

Discussion Question. Answer the fifth question below in a post to the Paper Discussion forum on Blackboard:

5. Müller et al. describe the Visibility technique and argue that it is modular. There is some reason to be skeptical however; for example consider the following text from a footnote: "declarations declared to be private will be visible but not accessible." Do you agree that the technique is modular, do you disagree, or do you partly agree? Explain.

After posting, you are invited (but not required) to read and respond to the comments of others in the course.