Research Topics in Software Quality

Assignment 3: Interval Data-flow Analysis

Academic Year 2014-2015
Coen De Roover, Quentin Stievenart
Vrije Universiteit Brussel

Submission deadline:
Sunday, 18 January 2015 by 23:59

Submit a written report about the following assignments by e-mail to cderoove@vub.ac.be. The report should be typeset in \LaTeX at a font size of 10pt on an A4 paper size. A minimum of 10 and a maximum of 30 pages is expected — not including appendices and references. Make sure each of the following points is discussed at length. Where possible, use small code snippets to illustrate your arguments. Include the complete source code for the assignment in a zip archive.

Discussion Point 1: Implementing an interval lattice
Implement a lattice of which the elements are intervals of integers (that is, \([n, m]\) where \(n\) and \(m\) are integers). Note that each bound of these intervals can be \(\infty\). Your lattice should support the \(\text{join} (\sqcup)\) and \(\text{meet} (\sqcap)\) operations, such that, for example:

\[
\begin{align*}
\{0, 10\} \sqcup \{15, 20\} &= \{0, 20\} \\
\{0, 10\} \sqcup \{5, 7\} &= \{0, 10\} \\
\{-\infty, 10\} \sqcup \{5, 15\} &= \{-\infty, 15\}
\end{align*}
\]

\[
\begin{align*}
\{0, 10\} \cap \{15, 20\} &= \bot \\
\{0, 10\} \cap \{5, 7\} &= \{5, 7\} \\
\{-\infty, 10\} \cap \{5, 15\} &= \{5, 10\}
\end{align*}
\]

Describe the top and bottom elements of this lattice, and explain how the meet and join operations work.

Also implement the four usual arithmetic operations \((+, -, \times, \div)\) on your lattice elements\(^1\).

\(^1\)Refer to the common definition of arithmetic operators for intervals: http://en.wikipedia.org/wiki/Interval_arithmetic
Discussion Point 2: Implementing the data-flow analysis  Implement the actual analysis that computes at each program point, for each integer variable, the interval corresponding to the possible values of this variable.

Explain whether the analysis is a forward or backward analysis and why.

Discuss the abstraction (i.e., the mapping from variables to intervals) to be computed by the data-flow analysis and how you have defined the merge operation on this abstraction.

Explain the initial value that you have used for this abstraction.

Finally, explain how the transfer function computes out values from in values. Implement the transfer function incrementally by handling, in increasing order of complexity, the following constructs:

• assignments to a constant,
• assignments to the result of an arithmetic operation (addition, subtraction, multiplication, division),
• arithmetic comparisons in branches (equality, non-equality, greater than, greater or equal, lesser than, lesser or equal).

After having implemented support for each of those constructs, show the corresponding computed values on the following functions, and explain how and why the analysis became more precise when handling each type of construct (assignment, arithmetic operations, branches). Use the following methods to illustrate your analysis:

```java
void foo(int x) {
    int a = 5;
    int b = a + 3;
    int c = b - 5;
    int d = c * 3;
    int e = d / c;
    int f = x + 5;
    // avoid e and f to be discarded by Java
    System.out.println(e);
    System.out.println(f);
}

int sign(int x) {
    int result;
    if (x == 0) {
        result = 0;
    } else if (x > 0) {
        result = -1;
    } else {
        result = 1;
    }
}


```java
return result;
}
```

**Discussion Point 3: Investigating loops** Run your analysis on the following method:

```java
int sum() {
    int i = 0;
    int sum = 0;
    while (i < 10) {
        sum += i;
        i++;
    }
    return sum;
}
```

If your analysis handles arithmetic and conditions, it shouldn't terminate on this method. Explain why by showing problematic values computed by your flowThrough method.

To solve this problem, analyses can use a technique called *widening*, which consists in intentionally losing precision on the values computed to accelerate the convergence to a fixed point. In the case of an interval analysis, a common technique is to widen each interval to the nearest enclosing interval made of constants present in the method (and infinity values). In the given method, the only constants that are present are 0 and 10, therefore the only possible intervals would be:

\[
\begin{align*}
\{-\infty, 0\}, & \{-\infty, 10\}, \{0, 0\}, \{0, 10\}, \{10, 10\}, \{0, +\infty\}, \{10, +\infty\}
\end{align*}
\]

To implement this, you will need to retrieve every integer constant appearing in the method. This can be done by looking for instances of `IntConstant` in the function body. Then, when creating an interval, use the nearest smaller constant (or \(-\infty\) if there are none) for the lower bound and the nearest bigger constant (or \(+\infty\)) for the upper bound.

Implement this technique, show the computed values for the methods `foo`, `sign`, and `sum`. Discuss the impact of widening on the precision of the analysis.