Automatic Software Verification

Ex. 2: Abstract Interpretation

Due 16/06/2015

This exercise is composed of 3 projects: you must do project 1, and may choose to do either project 2 or 3.

Project 1: "May-Be-Garbage" Analysis

The Python code used in class to demonstrate chaotic iteration and abstract interpretation can be found at:

https://bitbucket.org/tausigplan/asv15/src/master/demos/chaotic_iteration/

1. Implement the abstract domain for may-be-garbage analysis that was described in class in a file called "mbg.py"
2. Demonstrate the analysis on two interesting examples where the analysis gives a precise result, in files "mbg_prog1.py" and "mbg_prog2.py"
3. Create in "mbg_prog3.py" an example where the analysis loses precision, and gives a false alarm. This means the analysis indicates that some variable may be garbage, when in fact it is not garbage in any program execution.

Projects 2 & 3: TVLA

Download and install the TVLA 3 system from:
http://www.cs.tau.ac.il/~msagiv/courses/asv/tvla3_stable.zip

Solve one of the following projects using TVLA.

Project 2: **Proving Partial Correctness of a Simple Mark and Sweep Garbage Collection** (directory examples/gc)

1. Document the predicates and the actions in the tvp files for the Mark and Sweep example. In particular, explain the instrumentation predicates and their meanings
2. Remove the focus operations in actionst_set.tvp and study the resulting analysis.
3. What are the difficulties in extending the analysis to handle Garbage Collection Algorithms like Copy Garbage Collection (see http://www.cs.cornell.edu/courses/cs312/2003fa/lectures/sec24.htm) in which the garbage collector can mutate the heap.
4. (Bonus) Add actions for showing that the Mark phase must eventually terminate. One way to show that is by showing that the set of nodes reachable from the pointer variable x (used in the while loop condition in mark.tvp) decreases in every loop iteration .

Project 3: Proving Partial Correctness of Sorting Algorithm (directory examples/sll_sorting)

1. Document the predicates and the actions in the tvp files for the sorting example. In particular, explain the instrumentation predicates and their meanings.
2. Remove the focus operations in actions.tvp and study the resulting analysis.
3. Write an improved version of bubble sort in C called smart-bubble-sort which stops once the list is sorted and doesn't compare elements which are already in place (using linked lists and pointers). Then, convert it manually into tvp and run it. Study the results of the analysis.
4. (Bonus) Add actions for showing that the loops in insertion sort and bubble sort must eventually terminate. One way to show that is by showing that the set of nodes reachable from the temporary variable x decreases at every loop iteration.