Techniques for Improving Software Productivity

'19 Spring Semester

Homework Assignment 5: Abstract Interpretation

The Python code used in class to demonstrate chaotic iteration and abstract interpretation can be found at: https://bitbucket.org/tausigplan/chaotic-iterations/

The skeleton code for this assignment can be found at: https://bitbucket.org/tausigplan/soft-prod19/src/master/exercises/ex5/

In this exercise you are asked to implement the May Be Garbage analysis. The goal of the May Be Garbage analysis is to discover, at each program location, which program variables may contain garbage values. In the setting of this question, all program variables may initially contain garbage values. A variable is guaranteed to not contain garbage if it is assigned a constant (integer) value, or if it is assigned the value of an expression that is guaranteed to not be garbage.

A couple of examples to illustrate the expected behaviour:

- After the assignment `x := 5`, `x` will not contain a garbage value.
- After the assignment `x := y + z`, `x` will not contain a garbage value iff the expression `y + z` is not a garbage value. The expression `y + z` is not a garbage value if both `y` and `z` are not garbage. If one (or both) of `y` and `z` may hold garbage values, then `y + z` may hold a garbage value.

You may assume that programs contain only control flow (loops and conditions) and assignments.

1. Implement the abstract domain for may-be-garbage analysis. The code outline can be found in the file `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

Hint: When implementing the may-be-garbage abstract domain you may look for inspiration in both the constant propagation and available expression domains we've seen in class.

Submission Guidelines

You should submit 4 python files – `mbg.py`, `mbg_prog1.py`, `mbg_prog2.py` and `mbg_prog3.py`, as well as a pdf explaining your programs and the results of running abstract interpretation on all 3 programs.

The abstract domain you write should be entirely contained in the file `mbg.py`. You can add as many functions as you like, but **you should not change any of the skeleton code** that is in the file as it appears in the repository.

Note that the abstract domain you write in `mbg.py` will be tested against various program, not just the programs you submit, and should work correctly on all. You are encouraged to test your abstract domain on as many examples as you can think of.