

Homework III
CS 8803 : Languages and Compilers for Embedded Systems
Topic : Data-flow Analysis
Due Date : July 9th, Thursday, 6 pm
Total points : 100

Important Instructions:

1. The homeworks are non-collaborative. You are allowed to discuss the homework amongst yourself for clarification purposes only. We strongly encourage you to clarify things with the TA however. Any collaboration about the solution will be treated as plagiarism.
2. Keep your answers precise and try not to be lengthy.
3. When necessary make suitable assumptions, clearly stating and highlighting them.
4. Please turn in the homework on time – late home-works may not be accepted.

Question I [30 points] The reaching definition algorithm presented in the class is conservative for being safe. That is the algorithm finds a super-set of reaching definition. Show an example using control flow graph of at least 7 basic blocks where the algorithm finds some reaching definitions which can never reach some program points. Discuss how will you remedy this over-approximation.

Question II [30 points] Devise a modification of copy propagation algorithm to perform demand driven copy propagation. That is the algorithm is given a set of copies $\{ a=b, c=e, b=g \}$ along with their originating basic blocks and it will not perform global copy propagation but will perform only the propagation of these copies.

Question III [30 points] It is proposed to use the constant propagation optimization in a profile guided manner. That is consider a use point of a variable a (such as $\dots = a$) – if this is reached from frequent paths that bear the constant value of a then we still want to go ahead and substitute ‘ a ’ by the respective constant. This will however, make code unsafe when reached from other paths where ‘ a ’ is not a constant. Show how it is still possible to perform this optimization without compromising safety. Through an example, show how you will perform this optimization using at least 7 basic blocks.

Question IV [10 points] It is proposed to use the points-to(p) set information for reaching definition analysis. Points-to() set shows a set of possible objects to which a pointer can point. We saw in class the safety issue of reaching definitions which must make sure the set of definitions found by our algorithm should be a super-set of actual ones. Using this observation and the use of points-to() sets in revising kill() and gen() sets in reaching definition analysis, what can we say about the points-to() sets safety issues? That is should the points-to() found by our algorithm be a superset or a subset of actual locations where a pointer can point to?

