CS243 Homework 1

Winter 2018

Due: January 24, 2018

Directions:

- Complete the following problems and hand them in at the beginning of class with your name and SUID at the top.
- Remember to complete the corresponding Gradiance quizzes by the start of class on the due date. **There are no late days for Gradiance.**
- If you need to use one or more late days, hand in your assignment by 4:30pm Thursday or Friday in Gates 407. Slide it under the door if it is locked.
- This is an individual assignment. You are allowed to discuss the homework with others, but you must write the solution individually. If you look up any material in the textbook or online, you should cite it appropriately.

Problem 1. Indicate which of the following operators defines a semi-lattice. If so, define the top and bottom element of the associated semi-lattice, if they exist. If not, justify your answer by listing the properties that fail to hold.

1. Set intersection
2. Set difference
3. Component-wise minimum for tuples of natural numbers (partial order)
4. Lexicographic minimum for tuples of natural numbers (total order)
5. Addition over complex numbers
6. Arithmetic mean over real numbers
7. AND over booleans
8. XOR (exclusive OR) over booleans

Problem 2. True or False? Briefly justify your answer in 1 to 5 lines.

1. There is at most one fixed point solution for a dataflow problem.
2. There is always at least one fixed point solution for a monotone dataflow problem with finite descending chains.
3. A semi-lattice can have at most one bottom element.

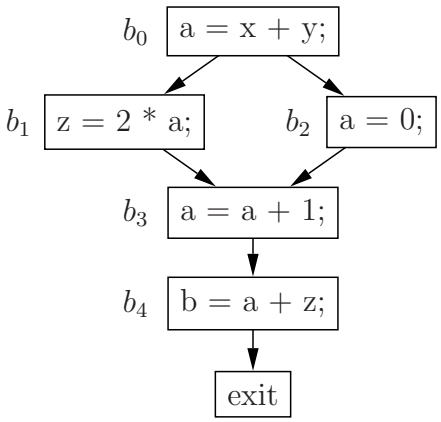
4. A bounded semi-lattice (one that has top and bottom) has necessarily a finite domain.

Problem 3. Live Range Analysis

We say that a path is *definition free* with respect to a variable y if there is no definition of variable y along that path. The live range of a definition $d : y = x + z$ that defines variable y consists of all the program points p such that both of the following hold:

1. There is a path from d to p that is definition free with respect to y.
2. There is a path from p to q that is definition free with respect to y, where q is a statement that uses the variable y.

Intuitively, the live range of a definition includes points along all subsequent paths until either it is overwritten by a new definition or the variable is no longer used along that path. The concept of live ranges can be used in register allocation: if live ranges of two definitions do not intersect, they can be assigned to the same register. Consider the following program:

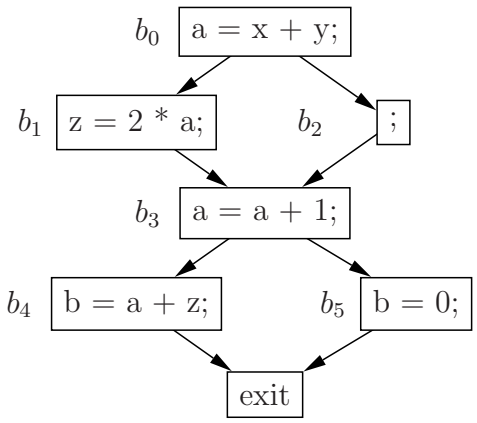


In this example, the live range of definition $a = x + y$ is exit(b_0), entry(b_1), exit(b_1) and entry(b_3). Similarly, the live range of the definition $b = a + z$ is exit(b_4), entry(EXIT). The two live ranges do not intersect, so b can reuse a's register.

Describe an analysis that computes the live range for each definition in a program. You may use algorithms discussed in class, in which case, you don't have to describe those algorithms again, but be specific on how you compute live range.

Problem 4. Detecting uninitialized variables.

Consider the following program:



There are no other instructions in this program. If the program follows the path b_0, b_1, b_3, b_4, the variable z is used without initialization. Conversely, if the program follows the path b_0, b_1, b_3, b_4, the variable z is initialized and then used. Unlike a program that always uses an uninitialized variable, this program is not necessarily incorrect: whether z is used uninitialized depends on the conditions on those branches. For this reason, compilers that warn about uninitialized variables usually have two options: one to warn about variables that are definitely uninitialized, and one to warn about variables that are maybe uninitialized (i.e. the compiler cannot prove that the variable is initialized).

In this problem, we'll consider how to implement those warnings using the dataflow framework.

1. What semi-lattice(s) can used to implement those warning passes? Define the domain, meet operator and bottom element.

2. What are the boundary conditions?

3. What are the initialization values for a maximum fixed point algorithm?

4. What is the transfer function on each instruction?

5. Once the dataflow algorithm completes, how do we find which instruction should generate a warning?

6. Would your dataflow passes warn on the program given above? If so, and assuming that the program was indeed correct, can you think of a refined algorithm which gives no warning?