Exercise 1: Due December 20

Note: The questions are specified in sloppy way to encourage you to formalize and prove the meaning of these questions and their answers.

Galois Connections

1. Prove that the two alternative definitions of Galois connections are indeed equivalent.
2. Let L be a lattice of abstract elements. Let \( \beta \): State \( \rightarrow \) L be the extraction function, i.e., \( \beta(\sigma) \) is the most precise conservative approximation of \( \sigma \) in L (as defined in class). Show the Galois connection from P(States) \( \rightarrow \) L induced by \( \beta \) and prove that it is indeed a Galois connection.
3. Show that the opposite direction of 3. also holds, i.e., for every Galois connection from P(States) to L there exists such an extraction function \( \beta \).

Pointer Analysis

1. Show that the abstract transformer of simple assignment \( x := y \) is indeed the best (induced)
2. Show that the abstract transformer of \( x := y \) is distributive (additive)
3. Is the abstract transformer of \( *x := y \) distributive?
4. Is the abstract transformer of \( *x := y \) the best one (induced)?
Interval Analysis

1. Show that the abstract meaning of the statement \( x := x + c \) in the interval analysis is the best (induced).

2. Generalize the domain of intervals to handle arbitrary number of program variables and show that it is a lattice and the generalized widening and narrowing. Define the Galois connection. Is it a Galois insertion?

3. Apply the Chaotic iteration algorithm with widening and narrowing to the C program and determine which of the array references to stack are guaranteed to be safe and what kind of runtime test is needed to guarantee safety (using the resultant intervals.)

4. Consider the following generalized C program which illustrates the usage of dynamic arrays in C (Java offers better facilities to define such arrays). Apply interval analysis to determine the potential values of the variable top. You can ignore statements that cannot affect the values of top.

   (Bonus) Develop an abstract domain which is precise enough to show that no array violations in C (and Java) programs with dynamic arrays. For simplicity, you can assume that the program manipulates a single array allocated using malloc with a designated variable, say size. Hint: one way to do that is to combine the lattice of signs with a domain which includes binary relationships between program variables. In the example program, we need to know that inside the print loop just before stack[i] is accessed, \( 0 \leq i < top \leq size \). This can be represented using three components:
   a. All the variables are positive
   b. The set \( \{(i, i), (i, top), (i, size), (top, top), (top, size), (size, size)\} \) which represents the inequalities:
   \[ i \leq i, i \leq top, i \leq size, top \leq top, top \leq size, size \leq size \]
   c. The set \( \{(i, top), (i, size)\} \) which represents the
inequalities: \( i < top, i < size \).
Define the domain and the Galois connection. Apply chaotic iterations to the example program with dynamic arrays.