CS615 Quiz 2

Max marks: 45 **Time: 1.5 hours**

- *Be brief, complete and stick to what has been asked.*
- *If needed, you may cite results/proofs covered in class without reproducing them.*
- ***If you need to make any assumptions, state them clearly.***
- ***Do not copy solutions from others.***

Consider the following program in the syntax studied in class. The program is annotated with location labels for ease of reference.

```
 0:  int x, y;

 1:  x := 0;
 2:  y := 0;
 3:  while (?) {
 4:     assume(x <= 10000);
 5:     x := x + 1;
 6:     y := y + 1;
 7:  }
 8:  assume(x > 10000);
 9:  if (?) {
10:    assume(y <= 0)
ERROR:  goto ERROR;
11:  }
```

We intend to determine whether location `ERROR` is reachable starting from the precondition `True`.

1. We will first use abstract interpretation in the domain of Difference Bound Matrices (DBMs) to analyze the above program.

   (a) *[5 marks]* Suppose we use the *lub* operator to estimate the loop invariant at location `3` after the first iteration through the loop, and use widen for all subsequent iterations. What are the estimates of loop invariants obtained at location `3` as the abstract analysis proceeds, and how many times do you need to iterate through the loop to determine that you have obtained a loop invariant in your abstract analysis?

   You need not list the DBMs obtained after every statement of the program as your analysis proceeds. It is sufficient to list the DBM at location `3` before/after every iteration through the loop until the loop invariant is obtained.

   (b) *[2 marks]* Does the loop invariant obtained above suffice to show unreachability of the `ERROR` location? Briefly justify your answer.

2. We now wish to use predicate abstraction using location specific predicates obtained as Craig interpolants from spurious counterexamples. We start off by not tracking any predicates at all.

   When a spurious counterexample is encountered, a simple strategy to compute Craig interpolants at different locations is as follows:

- **Strategy 1**: Let X_1, X_2, X_n be (potentially overlapping) sets of integer valued variables. Let $p_1(X_1), p_2(X_2), \ldots p_n(X_n)$ be quantifier-free predicates, where each predicate $p_i(X_i)$ expresses a linear equality or inequality between variables in X_i. For $1 \leq i \leq j \leq n$, we define $Y_{i,j} = \bigcup_{k=i}^{j} X_k$, $\psi_j^-(Y_{1,j}) = \bigwedge_{k=1}^{j} p_k(X_k)$ and $\psi_j^+(Y_{j+1,n}) = \bigwedge_{k=j+1}^{n} p_k(X_k)$. If $\psi_j^-(Y_{1,j}) \wedge \psi_j^+(Y_{j+1,n})$ is unsatisfiable, we compute a Craig interpolant between $\psi_j^-(Y_{1,j})$ and $\psi_j^+(Y_{j+1,n})$ by existentially quantifying out all variables in $Y_{1,j} \setminus Y_{j+1,n}$ from $\psi_j^-(Y_{1,j})$. This gives a conjunction of one or more equalities and/or inequalities between variables in $Y_{1,j} \cap Y_{j+1,n}$. We use each equality or inequality in the resulting interpolant as a predicate to track at the corresponding location of the program.

(a) *[5 marks]* Consider a spurious counterexample corresponding to following sequence of locations: `1, 2, 3, 8, 9, 10, ERROR`. This corresponds to hitting the `ERROR` location without entering the `while` loop even once, and can be obtained if we are not tracking any predicates at all. Using interpolating strategy 1 described above, compute predicates to be tracked at each location that appears in the counterexample. Your predicates should be such that tracking these predicates at the corresponding locations guarantees the elimination of this counterexample in the next iteration of predicate abstraction.

(b) *[10 marks]* Suppose after the above counterexample is eliminated, the abstract analysis indicates the presence of a counterexample corresponding to the following sequence of locations: `1, 2, 3, 4, 5, 6, 7, 3, 8, 9, 10, ERROR`. This corresponds to hitting the `ERROR` location after iterating through the `while` loop only once. It turns out that this is a spurious counterexample as well. Using the same interpolating strategy 1 described above, what are the final set of predicates to be tracked at each location of the program after the above two spurious counterexamples have been analyzed.

(c) *[3 marks]* If we continue the same process as above, i.e. identify abstract counterexamples of the ***shortest length*** every time a set of location specific predicates are used to do predicate abstraction, and analyze the counterexample to refine the abstraction, how many refinement iterations are needed before the CEGAR loop terminates? Briefly justify your answer.

(d) *[5 marks]* Suppose you had the ability to tell the predicate abstraction refinement tool that it must track a user-provided set of predicates at a specific location, in addition to whatever location specific predicates it automatically finds using Craig interpolation. Indicate what predicate(s) you would ask the tool to track and at which location such that the number of refinement iterations is minimized. Note that you are allowed to specify a set of predicate(s) for ***only a single location*** in the program.

(e) *[10 marks]* Briefly describe an alternnative strategy (different from that in strategy 1 above) for computing a Craig interpolant between $\psi_j^-(Y_{i,j})$ and $\psi_j^+(Y_{j+1,n})$ as described above. Your method/algorithm must not use anything other than existentially quantifying out variables from a conjunction/disjunction of linear equalities/inequalities and possibly negating linear equalities/inequalities. Briefly justify why your algorithm/method gives a Craig interpolant.

(f) *[5 marks]* Repeat part (a) of this question using your algorithm/method for computing location specific predicates.