CS615 Endsem Exam (Autumn 2016)

Max marks: 110 **Time: 180 mins**

- *Be brief, complete and stick to what has been asked.*
- *Unless asked for explicitly, 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. Penalty for offenders: FR grade.***

1. *This questions deals with the use of interval abstract domain and threshold widening.*

   Consider the following program P in a C-like language, where `bool foo(int j)` is an unspecified function that takes an integer argument and returns a boolean value.

```
L1: int j = 0;
L2: while ((j >= 0) && (j <= 5000)) {
L3:   if (j <= 1000) {
L4:     j = j + 2;
L5:   }
L6:   else {
L7:     if (foo(j)) {
L8:       j = j - 999;
L9:     }
L10:    if (j > 1500) {
L11:      j = j + 2000;
L12:      printf("Whoa!");
L13:    }
L14:  }
L15: }
```

   We wish to use the interval domain to answer the following two questions for this program.

   Q1: Does the program terminate?

   Q2: Does the program ever print "Whoa!"?

   Towards this end, we'll proceed step-by-step as below. :

   (a) *[10 marks]* Use the abstract domain of intervals with the usual widening operator for intervals studied in class to compute a loop invariant at `L1`. You may defer widening by atmost 2 iterations if you wish to. Clearly show the steps of your calculation.

(b) *[5+5 marks]* Using the loop invariant obtained in the previous sub-question, provide answers to Q1 and Q2. Using a wrong invariant from the previous question will lead to zero marks for this sub-question.

(c) *[5 marks]* Recall the notion of threshold widening studied in class. We wish to use a single threshold t to refine the widening operator for intervals for use in the above analysis, such that the answers to both Q1 and Q2 are "No". Indicate what is the smallest value of threshold t that can be used for this purpose.

(d) *[10+5+5 marks]* Re-do the calculations of sub-questions (a) and (b) above using threshold widening with the single threshold t, and show that this yields the answer "No" to both Q1 and Q2. Simply indicating a threshold without showing this calculation will fetch no marks for sub-questions (c) and (d).

2. *This question deals with the theory of abstract interpretation.*

Let (α, γ) be a pair of abstraction and concretization functions that form a Galois connection. Let the abstract domain under consideration be $\mathcal{A} = (A, \sqsubseteq, \sqcup, \sqcap, \top, \bot)$, where A is of infinite cardinality, and $\mathcal{A}$ is a complete lattice. The concrete domain is the usual $\mathcal{C} = (\wp(C), \subseteq, \cup, \cap, C, \emptyset)$, where C is the set of all concrete states of a program P. Assume that $\alpha(\emptyset) = \bot$, $\gamma(\bot) = \emptyset$, $\alpha(C) = \top$ and $\gamma(\top) = C$.

Let $F^{\mathcal{C}} : \wp(C) \to \wp(C)$ be a monotone concrete state transformer, and let $F^{\mathcal{A}} : A \to A$ be an abstract state transformer satisfying $\forall a \in A$, $\alpha(F^{\mathcal{C}}(\gamma(a))) \sqsubseteq F^{\mathcal{A}}(a)$. ***You are not allowed to make any other assumptions, viz. monotonicity, about*** $F^{\mathcal{A}}$***.***

(a) *[5 marks]* Is it necessary for $F^{\mathcal{A}}$ to have a fixed point in $\mathcal{A}$? Either give a proof of existence of a fixed point, or provide a counter-example.

(b) *[5 marks]* In case $F^{\mathcal{A}}$ has a fixed point in $\mathcal{A}$, does it necessarily have a least fixed point? Either prove that whenever $F^{\mathcal{A}}$ has fixed-point, it also has a least fixed point, or give an example where $F^{\mathcal{A}}$ has a fixed-point, but no least fixed point.

(c) *[10 marks]* By Knaster-Tarski Theorem, we know that $F^{\mathcal{C}}$ has a unique least fixed point. Denote this by μ. Now consider the sequence $a_0 = \bot$, and $a_{i+1} = F^{\mathcal{A}}(a_i)$ for all $i \geq 0$.

Does there always exist a natural number k such that $\mu \subseteq \gamma(a_k)$? Either give a proof, or provide a counter-example. Remember, you cannot assume monotonicity of $F^{\mathcal{A}}$ in this question.

3. *This question deals with predicate abstraction and Boolean programs.*

Consider the same C-like program given in question 1 of this paper. Let b_1 and b_2 be Boolean variables denoting the predicates $(j \geq 0)$ and $(j \leq 1000)$ respectively.

(a) *[10 marks]* Construct the best (i.e. preserving as much behaviour of the C-like program as possible) Boolean program as you can using the above Boolean variable. You must neatly present your Boolean program as the final answer, and also indicate how you obtained the various Boolean expressions used in your program. You may assume that the `printf` statement has no effect on the values of program variables.

(b) *[5 marks]* Give a trace, i.e. sequence of program locations, of your Boolean program that eventually reaches location `L12`, i.e. the `printf` statement in the original program.

(c) *[5 marks]* Construct the trace formula corresponding to the above trace and argue why it is unsatisfiable. You must give adequate justification for the unsatisfiability of the trace formula.

(d) *[10 marks]* For each location of the original C-like program that is present in the trace obtained in sub-question (b), derive predicates based on Craig interpolants such that a Boolean program constructed using your predicates does not permit the trace of sub-question (b) to arise. You must clearly show the Craig interpolant for each location, and the corresponding predicate derived from it.

4. *This question deals with separation logic formulas for representing heaps in program analysis.*

As discussed in class, a linked list pointed to by `x` and containing at least one element, can be defined using a recursive predicate in separation logic as follows:

$$list(x) = (x \mapsto NULL) \vee \exists t \ ((x \mapsto t) \star list(t))$$

Use separation logic to describe the post-condition as accurately as you can in each of the following Hoare triples:

(a) *[5 marks]* $\{list(x)\}$ `x := NULL;` $\{\ldots\}$

(b) *[5 marks]* $\{list(x)\}$ `x := *x;` $\{\ldots\}$

(c) *[5 marks]* $\{list(x) \star list(y)\}$ `temp := x; x := *y; free(temp);` $\{\ldots\}$