CS615 Midsem Exam (Autumn 2017)

Max marks: 58 **Time: 120 mins**

- *Be brief, complete and stick to what has been asked.*
- *Unless asked for explicitly, you can cite results/proofs covered in class.*
- ***If you need to make any assumptions, state them clearly.***
- ***Read the question paper carefully before answering questions.***
- ***Do not copy solutions from others. Penalty for offenders: FR grade.***

1. In class, we have always considered an abstract domain to be a lattice. In this question, we relax this requirement and consider an abstract domain $\mathcal{A} = (A, \leq, \widetilde{\sqcup}, \widetilde{\sqcap}, \top, \bot, \widetilde{\nabla})$, where

   - $\leq$ is a pre-order, i.e. a reflexive, transitive (not necessarily anti-symmetric) relation.
   - $\forall a \in A,\ \bot \leq a \leq \top$
   - $\forall a_1, a_2 \in A,\ a_i \leq a_1 \mathbin{\widetilde{\sqcup}} a_2$ for $i \in \{1, 2\}$ (i.e. $a_1 \mathbin{\widetilde{\sqcup}} a_2$ is some upper bound of a_1 and a_2). Also, $a_1 \leq a_2 \Rightarrow (a_1 \mathbin{\widetilde{\sqcup}} a_2) = a_2$.
   - $\forall a_1, a_2, a_3 \in A,\ (a_3 \leq a_1$ and $a_3 \leq a_2) \Rightarrow a_3 \leq (a_1 \mathbin{\widetilde{\sqcap}} a_2)$ (i.e. $a_1 \mathbin{\widetilde{\sqcap}} a_2$ is at least as large as any lower bound of both a_1 and a_2). Also, $a_1 \leq a_2 \Rightarrow (a_1 \mathbin{\widetilde{\sqcap}} a_2) = a_1$.
   - $\widetilde{\nabla}$ is a widening operator, i.e.
     - $\forall a_1, a_2 \in A,\ a_i \leq (a_1 \mathbin{\widetilde{\nabla}} a_2)$ for $i \in \{1, 2\}$.
     - For all non-decreasing chains $a_0 \leq a_1 \leq \ldots$ in A, the sequence $b_0, b_1, \ldots$ stabilizes after a finite number of terms, where $b_0 = a_0$ and $b_i = b_{i-1} \mathbin{\widetilde{\nabla}} a_i$ for $i \geq 1$.

     Also, assume that for all $c, d \in A,\ d \leq c \Rightarrow c \mathbin{\widetilde{\nabla}} d = c$.

   Let $\alpha : \wp(\mathcal{C}) \to A$ and $\gamma : A \to \wp(\mathcal{C})$ be a pair of abstraction and concretization functions that form a Galois connection, where $\mathcal{C}$ is the set of concrete states. Assume that $\gamma(\bot) = \emptyset$, $\gamma(\top) = \mathcal{C}$, $\alpha(\mathcal{C}) = \top$ and $\alpha(\emptyset) = \bot$.

   We wish to calculate the abstract loop invariant λ^A for "`while (B) P;`" using $\mathcal{A}$. Assume that `P` always terminates, and the set of concrete states reaching the loop-head for the first time is $\gamma(a_0)$, where $a_0 \in A$.

   Let $F^A : A \to A$ be an abstract state transformer (i.e. computes a post-condition, given a pre-condition) for the loop-body P, and let $F^C : \wp(\mathcal{C}) \to \wp(\mathcal{C})$ denote the concrete state transformer for P. Assume that (i) F^A is monotone with respect to $\leq$, (ii) $\forall a \in A,\ \alpha(F^C(\gamma(a))) \leq F^A$, and (iii) $F^A(\bot) = \bot$.

   A student uses the following procedure to try to calculate an abstract loop invariant.

   Step 1: Define $G : A \to A$ as $G(a) = a_0 \mathbin{\widetilde{\sqcup}} F^A(a \mathbin{\widetilde{\sqcap}} \alpha(B))$ for all $a \in A$.

Step 2: Compute $G^{10}(\bot)$, and check if $G^{11}(\bot) = G^{10}(\bot)$. If yes, report $\lambda^A = G^{10}(\bot)$.

Step 3: Otherwise, construct the following sequence $(\lambda_0, \lambda_1, \ldots)$ until $G(\lambda_i) \leq \lambda_i$.

Step 3a: $\lambda_0 = G^{10}(\bot)$

Step 3b: $\lambda_i = \lambda_{i-1} \mathbin{\widetilde{\nabla}} G(\lambda_{i-1})$ for $i \geq 1$

Step 4: If the sequence stabilizes at λ_m, report $\lambda^A = G^{10}(\lambda_m)$.

(a) *[10 marks]* Show that G is not necessarily monotone even when F^A is monotone. Specifically, give an example of $(A, \leq, \mathbin{\widetilde{\sqcup}}, \mathbin{\widetilde{\sqcap}})$, elements $a_0, a_1, a_2, b \in A$ and a function $F : A \to A$ such that

- F is monotone w.r.t. $\leq$
- $G(a) = a_0 \mathbin{\widetilde{\sqcup}} F(a \mathbin{\widetilde{\sqcap}} b)$ for all $a \in A$
- $a_1 \leq a_2$ and $G(a_1) \not\leq G(a_2)$.

(b) *[10 marks]* Show that *if the student's procedure terminates*, the computed λ^A satisfies the property that $\gamma(\lambda^A)$ is a concrete loop invariant.

(c) Unfortunately, the student is unable to show that her procedure always terminates. In a desperate bid to ensure termination, the student modifies Step 3b to the following:

Step3b*: $\lambda_i = \lambda_{i-1} \mathbin{\widetilde{\nabla}} F^A(\lambda_{i-1} \mathbin{\widetilde{\sqcap}} \alpha(B))$ for $i \geq 1$

i. *[5 marks]* Does the modified procedure always terminate?

ii. *[5 marks]* If the modified procedure terminates and returns λ^A, does $\gamma(\lambda^A)$ always represent a concrete loop invariant?

Justify your answer with explanation; simply saying "Yes"/"No" will fetch no marks.

2. We have studied in class how to model a single statement in a C-like programming language with possibly multiple statements in a Boolean program. Here, we wish to do the same, but with additional syntactic restrictions on the Boolean program statements.

Specifically, in each subquestion below, you are given a statement in a C-like programming language with all variables being of type `int`. You are also given a set of predicates, and a template of a *(single)* corresponding statement in the Boolean program. You are required to indicate the (possibly conditional) Boolean expression or $*$ that best replaces each E_i in the template, so that we get the best possible abstraction of the original program statement, given the syntactic restrictions imposed by the template.

Note that you are not allowed to add extra statements like `assume`. You **may not make any assumption** about the values of the Boolean variables b_i prior to the execution of the corresponding Boolean statement.

(a) *[8 × 2 marks]* **C-like program statement:** `x = (y > z) ? x-z : z-y;`
Predicates: $b_1 \equiv (y > z)$, $b_2 \equiv (x > y)$, $b_3 \equiv (x > z)$, $b_4 \equiv (x > 0)$
Boolean program statement template:
`(b1, b2, b3, b4) = b1 ? (E1, E2, E3, E4) : (E5, E6, E7, E8);`

(b) *[6 × 2 marks]* **C-like program statement:** `if (x > y) then y++; else x++;`
Predicates: $b_1 \equiv (x > y)$, $b_2 \equiv (y > 100)$, $b_3 \equiv (x < 100)$
Boolean program statement template:
`if (b1) then (b1, b2, b3) = (E1, E2, E3); else (b1, b2, b3) = (E4, E5, E6);`