Assignment 5: Termination, DPLL
15-414/15-424 Bug Catching: Automated Program Verification

Due: **11:59pm**, Thursday 10/12/17
Total Points: 50

1. **Find the variant (5 points)** In Homework 3, you proved the validity of the following formula which states partial correctness of the `gcd` program.

   $$0 < a \land 0 < b \to [c := a; d := b; \texttt{while}(c \neq d)\,\{\texttt{if}(c > d)\,c := c - d\,\texttt{else}\,d := d - c](c = gcd(a, b))$$

   Now give a variant term that is sufficient to prove the corresponding total correctness formula.

   $$0 < a \land 0 < b \to \langle c := a; d := b; \texttt{while}(c \neq d)\,\{\texttt{if}(c > d)\,c := c - d\,\texttt{else}\,d := d - c\rangle(c = gcd(a, b))$$

   You do not need to give a proof, but do concisely explain why your variant term is sufficient. You may refer to your invariant from homework 3 when justifying the invariant.

2. **More variants and invariants (5 points)** Consider the following program α, which finds the index of the maximum element of an array a of length n.

   ```
   i := 0;
   j := n-1;
   while(i ≠ j) {
     if(a(i) ⩽ a(j))
       i := i + 1;
     else
       j := j - 1;
   }
   ```

   Find a loop invariant and variant term sufficient to prove the validity of the formula:

   $$0 < n \to \langle\alpha\rangle\ (0 \leqslant i < n \land \forall k.0 \leqslant k < n \to a(k) \leqslant a(i))$$

   You do not need to show a full proof, but justify the correctness of your variant and invariant.

3. **LCP: specification (5 points)** Longest Common Prefix is an algorithm that is used in string search and other text problems. Given two indices i and j into an array, the goal is to compute the longest common prefix of the subarrays starting at i and j. That is, this returns the largest integer l such that the first l elements after indices i and j match.

   Consider the following program, which computes the longest common prefix of indices i and j for an array a of length n:

   ```
   l := 0;
   while(i+l < n ∧ j+l < n ∧ a(i+l) = a(j+l)
     l := l+1
   ```

   First, write down a precondition `pre` and postcondition `post` that specifies the partial correctness of this program. *Hint: As the postcondition may be long, you might find it helpful to factor the postcondition into predicates* $\texttt{cp}(a, i, j, l)$*, which specifies that the subarrays of* a *given by* i, j, l *are common prefixes, and* $\texttt{lcp}(a, i, j, l)$ *which uses* `cp` *to specify the longest common prefix.*

4. **LCP: variants and invariants (5 points)** Now provide a loop invariant and variant term that is sufficient to prove total correctness of the LCP program from problem 3 for the pre and postconditions you gave. You do not need to prove them yet, but briefly justify why they are correct.

5. **LCP: proof (10 points)** Finally, use the axioms of dynamic logic to prove total correctness of LCP using your specification from problem 3 and your annotations from problem 4. That is, prove the validity of the following formula:

$$\texttt{pre} \to \langle l := 0; \texttt{while}(i + l < n \wedge j + l < n \wedge a(i + l) = a(j + l))\ l := l + 1 \rangle\ \texttt{post}$$

6. **Soundness of $\mathrm{K}_{\langle\cdot\rangle}$ (5 points)** We discussed the modal modus ponens axiom $\mathrm{K}_{\langle\cdot\rangle}$:

$$(\mathrm{K}_{\langle\cdot\rangle}) \quad [\alpha](P \to Q) \to (\langle\alpha\rangle P \to \langle\alpha\rangle Q)$$

We used this to derive the inv2var rule, which is immensely useful as it allows us to use a previously-proved invariant property in our diamond proofs for convergence. Prove the soundness of $\mathrm{K}_{\langle\cdot\rangle}$.

7. **Pigeonhole SAT (10 points)** The pigeonhole problem asks us to find a one-to-one mapping between n pigeons and m holes. Obviously, this isn't possible when $n > m$. Consider an encoding of this problem as SAT for n pigeons and $n - 1$ holes, where we have the following CNF clauses and propositional variables p_{ij} which assert that pigeon i is placed in hole j.

   - *Pigeon clauses*: For each pigeon $0 \leqslant i \leqslant n$, assert that it is placed in some hole.

   $$p_{i,1} \vee \ldots \vee p_{i,n-1}$$

   - *Hole clauses*: For each hole $0 \leqslant j < n$ and each pair of pigeons $0 \leqslant i < k \leqslant n$, these two pigeons aren't placed in the same hole:

   $$\neg p_{i,j} \vee \neg p_{k,j}$$

   First, write down a CNF for the pigeonhole problem for $n = 3$. Then, apply the DPLL algorithm with clause learning to the formula. You should write down the steps of your evaluation in the following form:

   (a) Decide p
   (b) Unit propagate q from clause C_2
   (c) Decide $\neg r$
   (d) Unit propagate s from clause C_1
   (e) Conflicted clause C_1
   (f) Learn conflict clause $\neg p \vee r$
   (g) ...

   You are free to generate conflict clauses using any of the methods described in the lecture notes, but you may want to look at the next problem before choosing one.

8. **Resolving conflict (5 points)** Use the resolution rule to derive a proof that one of your conflict clauses from question 7 is entailed by the original formula.

$$(\text{res}) \quad \frac{\Gamma \vdash P, \Delta \quad \Gamma \vdash \neg P, \Delta}{\Gamma \vdash \Delta}$$

   As the formula is quite long, you may use the symbol F to denote the formula in your premises, so if your conflict clause is C, then you are to use the res rule to prove $F \vdash C$.

   How many applications of res were necessary in your proof? Do you think that it is possible to find a shorter one? Explain your answer.