Quiz

Synthesis, Analysis, and Verification 2011

Tuesday, March 29, 2010

Last Name : _____

First Name : _____

Exercise	Points	Achieved Points
Total	0	

Problem 1: Relations (10 points)

In this quiz we use the following notation:

- S is a set of all states
- $\mathsf{sp} : 2^S \times 2^{S \times S} \to 2^S$ is defined by

$$\mathsf{sp}(P, r) = \{s' \mid \exists s.s \in P \wedge (s, s') \in r\}$$

- $\mathsf{wp} : 2^{S \times S} \times 2^S \to 2^S$ is defined by

$$\mathsf{wp}(r, Q) = \{s \mid \forall s'.(s, s') \in r \to s' \in Q\}$$

- Diagonal relation Δ_A for $A \subseteq S$ and $r \subseteq S \times S$ is defined by

$$\Delta_A = \{(s, s) \mid s \in A\}$$

Task: prove the following equations:

a)

$$sp(P, r) = sp(S, \Delta_P \circ r)$$

b)

$$wp(r, S) = \{x | sp(\{x\}, r) \subseteq S\}$$

Problem 2: Transitive Closure by Fast Exponentiation (20 points)

Let A be a finite set containing exactly N elements ($|A| = N$). Let $r \subseteq A \times A$. Define a sequence of relations $r_1, r_2, \ldots$ using the following rules:

- $r_1 = r \cup \Delta_A$
- $r_{k+1} = r_k \circ r_k$, for $k \geq 1$

Task a): Prove that there exists $d > 0$ such that $r_k = r^*$ for all $k \geq d$. In other words, the sequence stabilizes with the transitive closure.
Task b): Find an estimate for d, i.e. a function $f(N)$ that grows more slowly than a linear function, such that $r_k = r^*$ for $k \geq f(N)$. The slower growing function, the better.

Problem 3: Formulas for Programs (35 points)

Consider the following code fragment:

```
q = 0;
i = 0;
while (i < M) {
  q = q + 3*i*(i+1) + 1;
  i = i + 1;
}
```

Assume that program state contains exactly the two variables, q, i, ranging over unbounded integers ($\mathbb{Z}$). M is a parameter that has some fixed but unknown positive integer value.

1. Guess a formula $F(x, y, x', y')$ describing (precisely) the execution of the entire program, between initial and final state of the program. To get the idea of what this program does, we suggest that you run it for a few iterations. Note that this program is deterministic, and your formula should uniquely specify the resulting values of integer variables after the loop execution finishes, as a function of parameter M.

2. Convert the program into guarded commands.

3. Compute the formula corresponding to this block iterated p times, that is,

   ```
   (
     assume(i<M);
     q = q + 3*i*(i+1) + 1;
     i = i + 1;
   )
   ```

   i.e. b^p where b is the relation for the body of the loop. You need to explain your steps, but there is no need to be very formal. To obtain a nicer solution, you may wish to use facts such as

   $$\sum_{k=0}^{p} k = \frac{p(p+1)}{2}$$

   $$\sum_{k=0}^{p} k^2 = \frac{p}{3}(p+1)(p+\frac{1}{2})$$

4. Use the formula from the previous part to compute the strongest postcondition formula of the program with respect to the precondition "true".

Problem 4: Hoare Triples and Loop Invariants (35 points)

Consider a programming language that supports integer variables, as well as variables that denote sets of integers and binary relations on integers (all integers are unbounded).
The command $\mathsf{lookup}(k, r)$ looks up a value v such that $(k, v) \in r$. If such value exists, it returns one such value as a singleton set $\{v\}$. If no such value exists, it returns the emptyset $\{\}$. (Note that, for each k, there can in general be zero, one, or more values v such that $(k, v) \in r$.)

Task a) Write a Hoare triple describing $\mathsf{lookup}(k, r1)$ in the form

$$\{\mathit{precondition}\} \quad v1 = \mathsf{lookup}(k, r1) \quad \{\mathit{postcondition}\}$$

where the precondition is as permissive (weak) as possible (so that it does not restrict the application of the lookup operation unnecessarily). Given as week precondition as you can find, specify the most precise postcondition that follows from the above description of how lookup should work.

Task b) Consider the following program, where the variables r1,r are relations, v1,W are sets of integers, and k is an integer.

```
// Precondition: ∀i.∀v.(i,v) ∈ r → 0 ≤ i
r1 = r;
k = 0;
W = {};
while // invariant Inv
       (r1 != {})
{
  v1 = lookup(k,r1);
  if (v1 = {}) {
    k = k + 1
  } else {
    W = W ∪ v1;
    r1 = r1 \ ({k} × v1)
  }
}
// Postcondition: W = range(r)
```

We use the notation

$$\mathsf{range}(r) = \{v \mid \exists i.(i, v) \in r\}$$

Find an appropriate loop invariant, Inv, and use it to prove that, whenever we run the above program in a state that satisfies the Precondition, its final state satisfies the Postcondition. You need to explain why (1) the invariant holds initially in *all* states that satisfy the precondition, why (2) it is inductive (preserved on each execution of the loop body starting from *any* state satisfying only the invariant), and why (3) it can be used to prove the Postcondition. State each of these conditions as a Hoare triple, and prove it. Your proof of individual Hoare triples need not be very detailed.
Feel free to use any notation of sets, relations, and quantifiers in your invariants and Hoare triples. It is crucial that your invariant is correct (conditions (1),(2),(3) hold). Hint: using $r \setminus r_1$ as part of your loop invariant may be helpful.