Quiz

Synthesis, Analysis, and Verification 2015

Wednesday, 22 April 2015

PLEASE SIGN AND PRINT YOUR NAME ABOVE

This exam has 5 questions.
When handing in, please hand in the sheets with questions as well as any additional sheets with solutions.

Problem 1: Relations ([14 points])

Suppose A is an arbitrary set of elements, and let $r \subseteq A \times A$ and $s \subseteq A \times A$ be two relations over A. For each of the following statements, determine whether it is true or not. If true, give a proof using known properties or definitions. If false, give a counterexample and show what each side evaluates to for these properties.
In addition to properties seen in class, you are allowed to use the following properties for your proofs:

- $(r^*)^* = r^*$
- $(r^* \circ r^*) = r^*$

Task a) (*4 points*)

$$(r \cup s)^* = r^* \cup (r \circ s)^* \cup (s \circ r)^* \cup s^*$$

Task b) (*4 points*)

$$(r \cup s)^* = (r^* \circ s^*)^*$$

Task c) (*3 points*)

$$(r \cap s)^* \subseteq r^* \cap s^*$$

Task d) (*3 points*)

$$r^* \cap s^* \subseteq (r \cap s)^*$$

Problem 2: Loop Semantics (hint: read the entire question first) ([30 points])

Assume m, n, x, y are the only global variables in the program, and that their values range over the integers $\mathbb{Z}$. The set of program states is $S = \{(m, n, x, y) \mid m, n, x, y \in \mathbb{Z}\}$. Consider the program:

```
while(x < n) {
  x = x + 1; y = y * m;
}
```

Task a) (*4 points*) Find the set $C \subseteq S$ and the relation $B \subseteq S \times S$ such that the meaning of the above program is given by the relation

$$(\Delta_C \circ B)^* \circ \Delta_{S \setminus C} \qquad (**)$$

where, for a set $A \subseteq S$, the diagonal Δ_A denotes $\{(t, t) \mid t \in A\}$. Specifically, give formulas C_F such that $C = \{(m, n, x, y) \mid C_F\}$ and B_F such that $B = \{((m, n, x, y), (m', n', x', y')) \mid B_F\}$.

Task b) (*9 points*)
In each of the following 3 cases, give a formula F that describes the program. Then indicate whether the value of $(**)$ above changes if we replace $\Delta_C \circ B$ with the relation $\{((m, n, x, y), (m', n', x', y')) \mid F\}$.

Task b.1) (*3 points*)

```
if(x < n) {
  x = x + 1; y = y * m;
} else {
  havoc(m,n,x,y);
}
```

Task b.2) (*3 points*)

```
if(x < n) {
  x = x + 1; y = y * m;
} else {
  assume(false)
}
```

Task b.3) (*3 points*)

```
if(x < n) {
  x = x + 1; y = y * m;
}
```

Task c) (*3 points*)
For each of the following initial states (m, n, x, y), compute $sp(\{(m, n, x, y)\}, r^5)$ where r is the meaning of program in **Task b.3)**.

- $(7, 2, 0, 1)$
- $(5, -2, 0, 1)$
- $(2, 3, -2, 2)$

Task d) (*6 points*)
Find (or guess) an exact formula L characterizing the entire loop, such that $(**)$ equals to the relation $\{((m, n, x, y), (m', n', x', y')) \mid L\}$. Your L should not contain any quantifiers, and may contain any well-known mathematical integer operations (not only linear arithmetic).

Task e) (*8 points*)
Write a precondition and postcondition for this program that is sufficient to prove uses of this code fragment for the purpose of computing exponentiation function with positive exponent. Find a loop invariant, prove it is correct, and use it to prove your postcondition for the code fragment.

Problem 3: Hoare Triples and Loop Invariants ([20 points])

Consider a programming language that supports integer variables, as well as variables that denote maps of integers to integers (all integers are unbounded). The maps are defined for all integers, meaning that for any map m and integer i, $\mathsf{m}(i)$ is well defined.
The command $r = \mathsf{m}(k)$ assigns the value of the map at index k to the variable r. The command $\mathsf{m}(k) = v$ update the map such that $\mathsf{m}(k)$ returns v.
We can use such a map type in combination with an integer variable length to model a regular array. The integer variable length gives the length of the array such that its elements are defined by $[\mathsf{m}(0), \mathsf{m}(1), \ldots, \mathsf{m}(\mathsf{length} - 1)]$. Note that the map m is still defined for integers outside the range of the array, and it is the job of the program to only interpret the relevant part of m.

Task a) (*5 points*)
Write a Hoare triple describing the $\mathsf{max}(\mathsf{m}, \mathsf{length})$ function that returns the max value of m between 0 and length. The max function is only defined on a non-empty array.

$$\{\textit{precondition}\}\ \ r = \mathsf{max}(\mathsf{m}, \mathsf{length})\ \ \{\textit{postcondition}\}$$

where the precondition is as permissive (weak) as possible (so that it does not restrict the application of the max operation unnecessarily). Given as weak precondition as you can find, specify the most precise postcondition that follows from the above description of how max should work. You can assume both m and length are immutable parameters and you do not need to model any potential updates.

Task b) (*15 points*)
Consider the following implementation of max:

```
def max(m, length) {
  i = 0;
  r = m(0);

  while // invariant Inv
        (i < length)
  {
    v = m(i);
    if (v > r) {
      r = v;
    }
    i = i + 1;
  }
  return r;
}
```

Find an appropriate loop invariant, Inv, and use it to prove that the max function meets the specification you found in **Task a)**. 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 the loop condition), and why (3) it can be used to prove the postcondition. State each of these conditions as a Hoare triple, and prove it. Your proofs of individual Hoare triples need not be very detailed, but they should give some explanations why these Hoare triples are mathematically true statements.

Problem 4: Lattices ([21 points])

In this example we consider finite lattices. If L is the set of elements of the lattice, then $|L|$ denotes the number of its elements.
Given a non-empty set I and a lattice $(L, \sqsubseteq)$, we define a new structure denoted $(L^I, \preceq)$ whose elements are functions $f : I \to L$ and whose ordering is defined by

$$(f \preceq g) \text{ if and only if } \forall i \in I.\ f(i) \sqsubseteq g(i)$$

Task a) (*9 points*)
Prove that $(L^I, \preceq)$ is a partial order and, moreover, a lattice and give definition for $\sqcup$ and $\sqcap$ on it. There is no need to assume that I is finite for this part.

Task b) (*2 points*)
Suppose that both L and I are finite. Then $(L^I, \preceq)$ is finite. Give the expression for the number of elements in it.

Task c) (*10 points*)
Suppose that both L and I are finite. If it helps you, you can assume that $I = \{1, 2, \ldots, K\}$.
The height of a lattice $(L, \sqsubseteq)$ is the length n of any of the longest chains $x_0 \sqsubset x_1 \sqsubset \ldots \sqsubset x_n$ of distinct linearly ordered elements. Here $x \sqsubset y$ means the conjunction of $x \sqsubseteq y$ and $x \neq y$. We use h to denote a function that computes the height of a lattice.
Give an expression for $h((L^I, \preceq))$ in terms of $|I|$ and $h((L, \sqsubseteq))$. Justify your response: describe one chain whose length equals to the claimed height, and also prove that there are no longer chains.

Problem 5: Predicate Abstraction ([15 points])

In this question we will consider bit-vector of size 32 typically used for the primitive integer type in programming language. Contrary to mathematical integers which are unbounded, bit-vectors are bounded and thus subject to overflow. Variables take a value in the set

$$\{-2^{31}, \ldots, -2, -1, 0, 1, 2, \ldots, 2^{31} - 1\}$$

Operations that would result in a value outside of this set are simply wrapped around, so that, for example, performing machine addition between the constant $2^{31} - 1$ and the constant 1 gives constant -2^{31}.
Consider the set of predicates over bit-vector variables

$$\mathcal{P} = \{\text{false}, 0 \le x, x < 0, 0 \le y, y < 0, x \le y, x \le 10, y \le 10\}$$

Let $A = 2^{\mathcal{P}}$.
For a given command `cmd` and $a \in A$, let $\mathsf{sp}^{\#}(a, \texttt{cmd})$ denote (as in the lecture) the conjunction of all predicates that are necessarily true after executing `cmd`, no matter which state satisfying the conjunction of the predicates a we start from:

$$\mathsf{sp}^{\#}(a, \texttt{cmd}) = \left\{P' \in \mathcal{P} \mid \ \{\bigwedge a\}\texttt{cmd}\{P'\}\ \right\}$$

Note that $\{\ldots\}c\{\ldots\}$ after "|" denotes a Hoare triple. Clearly $\mathsf{sp}^{\#} : A \times Commands \to A$.
Compute the following sets of predicates after the given commands (which, as given below, are either individual assignments of sequential compositions of assignments):

a) $\mathsf{sp}^{\#}(\{0 \le x, 0 \le y, x \le y\}, x = x + 1)$

b) $\mathsf{sp}^{\#}(\{0 \le x, 0 \le y, x \le 10, x \le y\}, (x = x + 1; x = x + 1))$

c) $\mathsf{sp}^{\#}(\mathsf{sp}^{\#}(\{0 \le x, 0 \le y, x \le 10\}, x = x + 1), x = x + 1)$

d) $\mathsf{sp}^{\#}(\{0 \le x, 0 \le y, x \le y\}, (x = x + 1; y = y + 1))$

Read the parentheses in the above expressions carefully.