CS 536: Quiz 2 (30 minutes) Solution

Instructions

The quiz is closed book, closed notes, and no support equipment (calculators, phones, computers, etc). All the questions are short-answer questions. The usual penalty for copying or sharing answers on a quiz or exam is a final grade of E for the course. If you have any questions, please ask during the quiz, not after. When a problem mentions "arbitrary" or "all" states, ignore states that improper (don't define enough values or have type-incorrect values).

Scores:

Histogram - with India Exams
100 100
99 99 99 99 99 98 98 97 97 96 96 95 95 95 95 95 95 95
94 94 94 94 92 92 92 92 92 92 92 91 91 91 91 91 90 90 90 90
89 89 88 88 87 87 87 86 86 86 85 85 85
84 84 84 83 82 82 82 81 81 80
79 79 77 76
75 74 73
66
54
Avg 88.6
Stdev 8.4

Questions

1. [16 points] Translate the C/C++/Java program fragment while $(x{+}{+} < y)$ { $y = ((y < 1)$? 0 : $x + y)$; } into an equivalent program in our simple programming language.

while $x < y$ **do**
  $x := x+1$;
  $y :=$ **if** $y < 1$ **then** 0 **else** $x+y$ **fi**
od;
$x := x+1$
(Note the increment of x after the loop; this handles the $x{+}{+}$ in test that breaks out of the loop.)

2. [30 points total] (a; 12 points) Let $S_1 \equiv x := x+1;\ y := y*x$. For all states σ, what is $\mathcal{M}[\![S_1]\!](\sigma)$? [Don't define specific values for $\sigma(x)$ and $\sigma(y)$; Your answer should work for all σ.] (*b*; 12 points) Let τ be a state that maps x, y, and n to 1, 1, and 3

respectively and let $W \equiv$ **while** $x < n$ **do** S_1 **od**. What sequence of states do we find at the loop test for W if we start in τ? (c; 6 points) What is the value of $\mathcal{M}[\![W]\!](\tau)$?

(2a) Let $\sigma' = \sigma[x := \sigma(x)+1]$ and $\sigma'' = \sigma'[y := \sigma(y) \times (\sigma(x)+1)]$, then $\mathcal{M}[\![S_1]\!](\sigma) = \{\sigma''\}$.
In detail, $\mathcal{M}[\![S_1]\!](\sigma) = \mathcal{M}[\![x := x+1;\ y := y*x]\!](\sigma)$

$= \mathcal{M}[\![y := y*x]\!](\mathcal{M}[\![x := x+1]\!](\sigma))$
$= \mathcal{M}[\![y := y*x]\!](\{\sigma'\})$
$= \{\sigma'[y := \sigma'(y*x)]\}$
$= \{\sigma'[y := \sigma'(y) \times \sigma'(x)]\}$
$= \{\sigma'[y := \sigma(y) \times (\sigma(x)+1)]\} \ = \{\sigma''\}$

(2b) $\tau_0 = \{\tau\}$, each $\tau_{j+1} = \mathcal{M}[\![S_1]\!](\tau_j) = \{\tau_j[x := \alpha_{j+1}][y := \beta_j\,(\alpha_{j+1})]$ where $\alpha_j = \tau_j(x)$ and $\beta_j = \tau_j(y)$. So $\alpha_{j+1} = \alpha_j+1$ and $\beta_{j+1} = \beta_j\,(\alpha_j+1)$. So $\alpha_0 = \beta_0 = 1$, $\alpha_1 = \beta_1 = 2$, $\alpha_2 = 3$, $\beta_2 = 6$.
(2c) We stop in state τ_2, so $\mathcal{M}[\![W]\!](\tau) = \{\tau_2\}$.

3. [3*10 = 30 points] Briefly describe the three trivial cases in which a partial correctness triple $\{p\}\ S\ \{q\}$ is valid (i.e., satisfied in all states)?

A short description of the cases is $\{\textbf{false}\}\ S\ \{q\}$, $\{p\}$ infinite loop $\{q\}$, and $\{p\}\ S\ \{\textbf{true}\}$. These triples are trivial because in the first case, S and q are irrelevant; in the second case, p and q are irrelevant; in the third case, p and S are irrelevant.

In more detail, (for all σ, $\sigma \models \{p\}\ S\ \{q\}$) is implied by each one of the following three:

- for all σ, $\sigma \not\models p$ (i.e., p is a contradiction).
- for all σ, $\mathcal{M}[\![S]\!](\sigma)$ is empty (i.e., S always diverges).
- for all σ, $\sigma \models q$ (i.e., q is a tautology).

A number of people tried to say "for all σ, $\mathcal{M}[\![S]\!](\sigma) \models q$". This is not a trivial situation; it says that running S establishes q regardless of p. E.g., $\models \{x > 0\}\ x := 2\ \{x > 1\}$.

4. [3*8 = 24 points] Let $p \equiv x > 1 \wedge k > 1 \rightarrow \exists y : 1 < y < x^k$, where x, y, and k are integer variables. (a) What is $p[x := k]$? (b) What is $p[x := y]$? (c) What is $p[y := x]$? (For all 3 parts, just calculate the substitution; don't logically simplify the result.)

(4a) $p[x := k] \equiv k > 1 \wedge k > 1 \rightarrow \exists y : 1 < y < k^k$. Many people wrote just $k > 1$ instead of $k > 1 \wedge k > 1$. Remember, "$\equiv$" is basically a string comparison. So $k > 1$ and $k > 1 \wedge k > 1$ are logically equivalent (so they're $\leftrightarrow$), but they're not syntactically equivalent (so they're not $\equiv$).

(4b) $p[x := y] \equiv y > 1 \wedge k > 1 \rightarrow \exists z : 1 < z < y^k$. You must change the bound variable, otherwise you capture y and get get $\exists y : 1 < y < y^k$. (You don't have to use "z", but you need something other than x, y, or k. Some people tried to write the change in bound variable as $p[y := z][x := y]$. This is incorrect because $p[y := z] \equiv p$ (see part 4c below). If you want to see the details (you didn't have to write all this), here they are. (Below, let's treat $\exists$ as having lower precedence than substitution.)

$$
\begin{aligned}
& p[x := y] \\
& \equiv (x > 1 \wedge k > 1 \rightarrow \exists y : 1 < y < x^k)[x := y] \\
& \equiv y > 1 \wedge k > 1 \rightarrow (\exists y : 1 < y < x^k)[x := y] \\
& \equiv y > 1 \wedge k > 1 \rightarrow \exists z : (1 < y < x^k)[y := z][x := y] \\
& \equiv y > 1 \wedge k > 1 \rightarrow \exists z : (1 < z < x^k)[x := y] \\
& \equiv y > 1 \wedge k > 1 \rightarrow \exists z : 1 < z < y^k.
\end{aligned}
$$

(4c) $p[y := x] \equiv p$. The occurrences of y in p are bound, so substituting x for the free occurrences of y changes nothing. Some people wrote that $p[y := x]$ is impossible; this is incorrect.