Forward Assignment; Strongest Postconditions

CS 536: Science of Programming, Spring 2022

For Questions 2 - 7, syntactically calculate the following, including intermediate *sp* calculation steps. Omit uses of "T ∧ " in the calculations but don't otherwise simplify the result unless asked to.

2. $sp(y \geq 0, skip)$
3. $sp(i > 0, i := i+1)$ [Hint: add an $i = i_0$ conjunct to $i > 0$]
4. $sp(k \leq n \wedge s = f(k, n), k := k+1)$
5. $sp(T, i := 0; k := i)$
6. $sp(i \leq j \wedge j\text{-}i < n, i := i+j; j := i+j)$.
7. $sp(0 \leq i < n \wedge s = sum(0, i), s := s+i+1; i := i+1)$
8. Let $S \equiv$ *if* $x < 0$ *then* $\{x := -x\}$ *else* $\{skip\}$
   a. Calculate $sp(x = x_0, S)$.
   b. Logically simplify your result from part (a). Feel free to use the function *abs(…)* or |…|.
   c. Suppose we had calculated $sp(T,$ *if* $x < 0$ *then* $\{x := -x\}$ *else* $\{skip\})$ introducing x_0 in the true branch only. What would we get for the *sp* and what is the problem with it?

Solution to Practice 13 (Forward Assignment; Strongest Postconditions)

2. $y \geq 0$ (For the *skip* rule, the precondition and postcondition are the same.)

3. Let's implicitly add $i = i_0$ to the precondition, to name the starting value of i. Then

   $sp(i > 0, i := i+1)$

   $\equiv (i > 0)[i_0/i] \wedge i = (i+1)\ [i_0/i]$

   $\equiv i_0 > 0 \wedge i = i_0+1$

4. As in the previous problem, let's introduce a variable k_0 to name the starting value of k. Then

   $sp(k \leq n \wedge s = f(k, n), k := k+1)$

   $\equiv (k \leq n \wedge s = f(k, n))[k_0/k] \wedge k = (k+1)[k_0/k]$

   $\equiv k_0 \leq n \wedge s = f(k_0, n) \wedge k = k_0+1$

5. We don't need to introduce names for the old values of i and k (they're irrelevant).

   $sp(T, i := 0; k := i)$

   $\equiv sp(sp(T, i := 0), k := i)$

   $\Leftrightarrow sp(i = 0, k := i)$ // We've dropped the "$T \wedge$ " part of $T \wedge i = 0$)

   $\equiv i = 0 \wedge k = i$

6. Let's introduce i_0 and j_0 as we need them, then

   $sp(i \leq j \wedge j\text{-}i < n, i := i+j; j := i+j)$

   $\equiv sp(sp(i \leq j \wedge j\text{-}i < n, i := i+j), j := i+j)$

   $\equiv sp(i_0 \leq j \wedge j\text{-}i_0 < n \wedge i = i_0+j, j := i+j)$

   $\equiv i_0 \leq j_0 \wedge j_0\text{-}i_0 < n \wedge i = i_0+j_0 \wedge j = i+j_0$

7. $sp(0 \leq i < n \wedge s = sum(0, i), s := s+i+1; i := i+1)$

   $\equiv sp(sp(0 \leq i < n \wedge s = sum(0, i), s := s+i+1, i := i+1)$

   For the inner *sp*,

$sp(0 \leq i < n \wedge s = sum(0, i), s := s+i+1)$

$\equiv 0 \leq i < n \wedge s_0 = sum(0, i) \wedge s = s_0+i+1$ Using s_0 to name the old value of s

Returning to the outer *sp*,

$sp(sp(0 \leq i < n \wedge s = sum(0, i), s := s+i+1), i := i+1)$

$\equiv sp(0 \leq i < n \wedge s_0 = sum(0, i) \wedge s = s_0+i+1, i := i+1)$

$\equiv 0 \leq i' < n \wedge s_0 = sum(0, i') \wedge s = s_0+i'+1 \wedge i = i'+1$

Using i' to name the old value of i

(There's no particular reason I used i' here except that; any other name like i_0 or j or w works fine as long as it's not already being used in the predicate.)

8. (Old value before an *if-else*)

a. $sp(x = x_0, \text{if } x < 0 \text{ then } \{x := -x\} \text{ else } \{skip\})$

$\equiv sp(x = x_0 \wedge x < 0, x := -x) \vee sp(x = x_0 \wedge x \geq 0, skip)$

$\equiv (x_0 < 0 \wedge x = -x_0) \vee (x \geq 0 \wedge x = x_0)$

b. We can simplify $(x_0 < 0 \wedge x = -x_0) \vee (x \geq 0 \wedge x = x_0) \Leftrightarrow x = |x_0|$.

c. If we had calculated

$sp(T, \text{if } x < 0 \text{ then } \{x := -x\} \text{ else } \{skip\})$

$\equiv sp(T \wedge x < 0, x := -x) \vee sp(T \wedge x \geq 0, skip)$

$\equiv (x_0 < 0 \wedge x = -x_0) \vee x \geq 0$

Then we would have lost the information about the *else* clause not changing x, so we wouldn't have been able to conclude $x = |x_0|$.