Universitetet i Oslo
Institutt for Informatikk
PMA
Olaf Owe, Martin Steffen

INF 4140: Models of Concurrency

Høst 2015 **Series 5** 12. 10. 2015

Topic: Program Analysis I

Issued: 12. 10. 2015

Exercise 1 (Substitutions) Do the following substitutions:

1. $(x = 2)[y/x]$
2. $(x = 2)[x + 2/x]$
3. $(x > y \wedge y \leq 2)[z + 2/y]$
4. $(x = 2)[2/x]$
5. $(x = y)[y + 1/x]$

Exercise 2 (The assignment axiom)

1. The following triples are instances of the assignment axiom:

$$\{\ 0 = 0 \wedge y < z\ \}\ x := 0\ \{\ x = 0 \wedge y < z\ \} \tag{1}$$
$$\{\ y - (x + 1) < z\ \}\ x := x + 1\ \{\ (y - x) < z\ \} \tag{2}$$

   Use [1, Definition 2.4 (p. 59)] to convince yourself about the "true-ness" of these triples in our computational model.

2. Given the following triples:

$$\{\ y < z\ \}\ x := 0\ \{\ x = 0 \wedge y < z\ \} \tag{3}$$
$$\{(y - x) = z\}\mathtt{x} := \mathtt{x} + \mathtt{1}\{y - x < z\} \tag{4}$$

   Convince yourself about the "true-ness" of these triples and use the axiom of assignment to *prove* them.

3. Consider the following two triples:

$$\{\ a = 2^k * y\ \}\ k := k - 1\ \{\ a = 2^{k+1} * y\ \} \tag{5}$$
$$\{\ a = 2^{k+1} * y\ \}\ y := y * 2\ \{\ a = 2^k * y\ \}\ . \tag{6}$$

   Again, use Definition 2.4 from the book to convince yourself about the trueness of these triples. Use the assignment axiom to prove their correctness. Can we use these two triples to say something about the program $k := k - 1; y := y * 2$?

4. The partial correctness interpretation of triples from Definition 2.4 from [1] expresses that $\{\ P\ \}\ S\ \{\ Q\ \}$ is true if, for *each* possible execution of S, starting in a state satisfying P, *if* the execution terminates at the end of S, then the program is in a state satisfying Q.[1] Thus, in order to prove that $\{\ P\ \}\ S\ \{\ Q\ \}$ is *not* true, it is enough to find one execution of S starting in a state satisfying P, but terminating in a state *not* satisfying Q. This execution then serves as a *counterexample* or witness for the true-ness of the triple. So: find counterexamples to show that the following two triples are not true:

$$\{\ x < 5\ \}\ \mathtt{x} := \mathtt{x} + \mathtt{1}\ \{\ x < 5\ \} \tag{7}$$
$$\{\ x \neq y\ \}\ \mathtt{x} := \mathtt{2} * \mathtt{x}\ \{\ x \neq y\ \} \tag{8}$$

Try to apply the axiom of assignment on the two triples. Are we able to prove these triples using the assignment axiom?

Exercise 3 (Free variables) The book [1, page 60] defines[2] $P[e/x]$ as P with all *free* occurrences of x replaced by e. An variable *occurs* in an expression/formula etc, if it "shows up" inside somewhere. A *free occurence* of a variable is an occurence of that variable which is not in the scope of a quantifier $(\forall, \exists)$. For instance, y occurs *free* in the predicate $\forall x : x \leq y$, but x does not occur free. Note that a variable may occur more than once in a formula, and hence may occur both free and non-free (one says also "bound") in one formula.

1. What is the result of the two substitutions
   (a) $(\forall x : x \leq y)[e/x]$ and
   (b) $(\exists x : x \leq y)[e/y]$?
2. Given the predicates:

$$P\colon\quad y > z \wedge (\forall i : 1 \leq i \leq n : a[i] <= max)$$
$$Q\colon\quad i > z \wedge (\exists i : 1 \leq i \leq n : a[i] <= max)$$

   Give the result of the substitutions $P[z{+}1/z]$, $P[max{*}2/max]$, and $P[j/i]$. Evaluate further $Q_{i \leftarrow j}$.

Exercise 4 (Swapping integers)

1. Imagine that we are trying to write a program that swaps the values of x and y without using any additional variables. We are not sure what the program should look like, and try three different suggestions:

$$\mathtt{x} := \mathtt{x} - \mathtt{y};\ \mathtt{y} := \mathtt{x} + \mathtt{y};\ \mathtt{x} := \mathtt{y} - \mathtt{x} \tag{9}$$
$$\mathtt{x} := \mathtt{y} - \mathtt{x};\ \mathtt{y} := \mathtt{y} - \mathtt{x};\ \mathtt{x} := \mathtt{x} - \mathtt{y} \tag{10}$$
$$\mathtt{x} := \mathtt{x} + \mathtt{y};\ \mathtt{y} := \mathtt{x} - \mathtt{y};\ \mathtt{x} := \mathtt{x} - \mathtt{y} \tag{11}$$

[1] Compared to Definition 2.4 of the book, I added terminates *at the end of S*. This is added for clarification. If S runs for example into a deadlock or gets blocked otherwise, then S may never reach the end of its execution. In that situation, the post-condition *is not* required to hold afterwards, since S has not reached the post-state; it's stuck somewhere in the middle. Nonetheless, one may may, informally, think that "S terminates" (in that it deadlocks), but that's not what is meant by the definition.

[2] Actually, everyone else defines substitution this way as well, it's the universally accepted definition.

Use *programming logic* (PL) to find out which suggestion(s) actually swap(s) the values of x and y. Given the precondition P as $\{\ x = x_0 \wedge y = y_0\ \}$, where x_0 and y_0 are *logical* variables. Find out if Q as $\{\ y = x_0 \wedge x = y_0\ \}$ holds upon termination in each of the three cases.

2. Given the followng program S:

$$\texttt{if(x} > \texttt{y)\{x} := \texttt{x} + \texttt{y; y} := \texttt{x} - \texttt{y; x} := \texttt{x} - \texttt{y\}} \tag{12}$$

Prove that

$$\{\ \mathit{true}\ \}\ S\ \{\ x \leq y\ \} \tag{13}$$

is a theorem (i.e., is derivable) in PL.

Exercise 5 (If-else rule) The book had given a derivation rule for a specific form of a conditional statement, namely one with only one branch. A reasoning rule for another conventional form of conditional statement, those with 2 branches, can be given as follows:

$$\frac{\{\ P \wedge B\ \}\ S_1\ \{\ Q\ \} \qquad \{\ P \wedge \neg B\ \}\ S_2\ \{\ Q\ \}}{\{\ P\ \}\ \texttt{if}\ (B)\ S_1\ \texttt{else}\ S_2\ \{\ Q\ \}}\ \textsc{Cond} \tag{14}$$

Compare this rule to the reasoning rule for if statements given in the lecture:

$$\frac{\{\ P \wedge B\ \}\ S_1\ \{\ Q\ \} \qquad P \wedge \neg B \Rightarrow Q}{\{\ P\ \}\ \texttt{if}\ (B)\ S_1\ \texttt{else}\ S_2\ \{\ Q\ \}}\ \textsc{Cond}' \tag{15}$$

Use the two-armed rule in (14) above to prove the following triple:

$$\begin{array}{l}\{\ x > 0 \wedge y > 0 \wedge x \neq y \wedge x + y = a\ \} \\ \texttt{if}\ (\texttt{x} < \texttt{y})\ \texttt{y} := \texttt{y} - \texttt{x}\ \texttt{else}\ \texttt{x} := \texttt{x} - \texttt{y} \\ \{\ x > 0 \wedge y > 0 \wedge x + y < a\ \}\end{array} \tag{16}$$

Exercise 6 (Exercise 2.20 from the book)
Let $a[1:m]$ and $b[1:n]$ be integer arrays, $m > 0$ and $n > 0$. Write predicates to express the following properties.

1. All elements of a are less than all elements of b
2. Either a or b contains a single zero, but not both.
3. It is not the case that both a and b contain zeros.
4. The values in b are the same as the values in a, except that they are in reverse order. (Assume for this part that $m = n$.)
5. Every element of a is an element of b.
6. Some element of a is larger than some element of b and vice versa.

References

[1] G. R. Andrews. *Foundations of Multithreaded, Parallel, and Distributed Programming.* Addison-Wesley, 2000.