

CSI 5110 Principles of Formal Software Development Fall 2011

Assignment 4

Assigned November 14, due Monday, November 28 at 8:30

November 24: Error corrected in last line of (r_1) action at the end of page 2.

1. p. 157, 2.1 (3a,b,c)
2. Assume that the substitution $A[t/x]$ is permitted only if t is free for x in A . Assume also that you can always rename bound variables so that t becomes free for x in A . Then after renaming you can then do the substitution (as illustrated in class). Let A be the following formula:

$$\forall y (P(x, y) \rightarrow Q(y)) \rightarrow \forall z (\neg P(x, z) \vee Q(z)).$$

- (a) Is $g(y, z)$ free for x in A ?
 - (b) Is $g(w, z)$ free for x in A ?
 - (c) Is $g(w, z)$ free for y in A ?
 - (d) What is the formula obtained by doing the substitution $A[g(w, z)/x]$?
3. p. 160, 2.3 (1c) You are not allowed to use the “arith” rule for problems in Chapter 2. Instead, use the equality rules on pages 107–108.
 4. p. 161, 2.3 (7c)
 5. p. 161, 2.3 (9o)
 6. Choose one of the sequents from question (3), (4), or (5) and do a proof in Coq of the sequent. Your Coq proof does not have to “match” your paper proof. You may use any Coq commands learned in class except for “auto” and “tauto”. Your solution must be sent to me by email, in a file named LastNameFirstInitial.v. I must be able to save the file, load it into Coq, and successfully execute every line. All solutions must be submitted before class on the due date.

7. Consider the SQR protocol discussed in class. Suppose we want to change the way we model this protocol so that the sender s has a buffer size of 2. The other two processes have a buffer size of 1 just as they did before. Assume that the buffer is first-in first-out. In other words, when a message is sent to s , if s 's buffer is empty, the message goes into the first slot in the buffer. If there is already one message, the message goes into the second slot. When s receives a message, it always receives it from the first slot in the buffer. If there are two messages, the one in the second slot moves to the first slot after s reads a message. We could model a state as before, with only a slight change (shown in bold) to the meaning of sc as below.

$$state(s, rc, \mathbf{sc}, qc, rd, qd, i, rec, sent)$$

- s : the current state number
- rc : value 1 if r 's channel is full, 0 otherwise
- \mathbf{sc} : value 1 if there is one message in s 's buffer, value 2 if s 's buffer is full, 0 otherwise
- qc : value 1 if q 's channel is full, 0 otherwise
- rd : value of argument to r send message in the r 's buffer if there is one, doesn't matter otherwise
- qd : value of first argument to q send message in q 's buffer if there is one, doesn't matter otherwise
- i : value of second argument to q send message in q 's buffer if there is one, doesn't matter otherwise
- rec : current value of variable rec
- $sent$: current value of variable $sent$

We now have to change the way we model the s_d action which takes a message out of s 's buffer, and the r_1 action which sends a message to s . Below are these two actions in the guarded command language taken directly from the class notes, followed by the original versions of the predicate logic formulas that specify them, as given in class.

$$\begin{aligned}
& (r_1) \text{ true} \longrightarrow s!request \\
& (s_d) \text{ buf}[s]?request \longrightarrow sent = d; q!qsend(d, 0) \\
& (r_1) \forall s \forall rc \forall qc \forall rd \forall qd \forall i \forall rec \forall sent \\
& \quad (state(s, rc, 0, qc, rd, qd, i, rec, sent) \rightarrow \\
& \quad \quad \exists s'(state(s', rc, 1, qc, rd, qd, i, rec, sent) \wedge r1(s, s'))) \\
& (s_d) \forall s \forall rc \forall rd \forall qd \forall i \forall rec \forall sent \forall d \\
& \quad (state(s, rc, 1, 0, rd, qd, i, rec, sent) \rightarrow \\
& \quad \quad \exists s'(state(s', rc, 0, 1, rd, d, 0, rec, d) \wedge sd(s, s')))
\end{aligned}$$

- (a) Give new predicate logic formulas for these two actions that use the new meaning of the $state$ predicate described above.
- (b) Express the following property in predicate logic: it is possible to get to a state where the sender has a backlog of 2 pending requests, but the receiver has no further data to process and there is no data in the queue.

- **Optional Extra Credit:** Prove the property you gave as an answer to question 7(b) above in Coq. Add your solution to the end of the file you create for Question 6 above.