Thinking Programs: Exercises

Chapter 9: Concurrent Systems

1. Take the following composition of two parallel processes P and Q operating asynchronously on shared variables a, b, c where the first process cycles among program counters $P_1 \to P_2 \to P_1 \to \ldots$ and the second one among $Q_1 \to Q_2 \to Q_1 \to \ldots$.

```
var a = 0, b = 0, c = 0
loop           ||  loop
  P1: a = a+c  ||    Q1: choose b = 0 | b = 1
  P2: c = c+b  ||    Q2: b = b+c
```

   Here the command `choose` c_1 `|` c_2 represents the non-deterministic choice between commands c_1 and c_2.

   First give a formal model of the system (using the interleaving assumption for asynchronous composition) as a labeled transition system with five transitions labeled `P1`, `P2`, `Q1a`, `Q1b`, and `Q2`; do not forget the definition of the state space. Then model this system as a "shared system" in the language presented in this book.

   Finally formalize in LTL the properties

   - "a stays zero forever".
   - "b becomes infinitely often one".
   - "a is zero until b is one (which may never be the case)".

   Which of these properties are true without fairness requirements (if a property is not true, show a violating system run)? Which do become true if we assume weak fairness for all transitions? Which do become true if we assume strong fairness for all transitions? Explain your answers.

2. Take the following asynchronous composition of two processes P, Q operating on shared variables a and b where the first process cycles among program counters $P_1 \to P_2 \to P_3 \to P_1 \to \ldots$ and the second one among $Q_1 \to Q_2 \to Q_1 \to \ldots$.

```
var a = 0, b = 0
loop              || loop
  P1: wait b != 0 ||   Q1: choose b = 1 | b = 2
  P2: a = b       ||   Q2: wait b == 0
  P3: b = 0       ||
```

   Here the `choose` command is defined as in Exercise 1. The command `wait` F blocks the execution of the process unless the condition F is true.

   First give a formal model of the system (using the interleaving assumption for asynchronous composition) as a labeled transition system with six transitions labeled `P1`, `P2`, `P3`, `Q1a`, `Q1b`, and `Q2`; do not forget the definition of the state space. Then also model this system as a "shared system" in the language presented in this book.

   Finally formalize in LTL the properties

- "b becomes infinitely often one".
- "always if b becomes one, then also a will become one".
- "b must become one before a can become one (which may never be the case)".

Which of these properties are true without fairness requirements (if a property is not true, show a violating system run)? Which do become true if we assume weak fairness for all transitions? Which do become true if we assume strong fairness for all transitions? Explain your answers.

3. Consider Peterson's algorithm that ensures that only one of two processes can enter a critical region:

```
flag[0] = flag[1] = 0
for i in in 0,1 do in parallel:
  loop
    flag[i] = 0
    turn = 1-i
    while flag[1-i] = 1 ∧ turn = 1-i do { }
    // critical region
    flag[i] := 0
```

First model the system consisting of two processes $i \in \{0, 1\}$ as a labeled transition system. Then also model this system as a "shared system" in the language presented in this book.

Finally formalize in LTL the properties:

a) Never both processes are in the critical region.

b) Every process infinitely often enters the critical region.

Which of these properties are true without fairness requirements (if a property is not true, show a violating system run)? Which do become true if we assume weak fairness for all transitions? Which do become true if we assume strong fairness for all transitions? Explain your answers.

4. Consider the alternating bit protocol that ensures the reliable transition of a sequence of messages from the sender to the receiver via a couple of shared variables:

```
sbit = sack = rbit = 0 // sent and rcvd are arbitrary
Sender:
  loop
    wait sack = sbit
    sent = any message
    sbit = 1-sbit
Receiver:
  loop
    wait rbit != sbit
    rcvd = sent
    rbit = sbit
    sack = rbit
```

First model this system as a labeled transition system (which of above assignments can you combine into a single transition without affecting the correctness of the protocol?). Then also model this system as a "shared system" in the language presented in this book.

Finally formalize in LTL the properties:

a) Every received message was sent.

b) Every sent message will be received.

Which of these properties are true without fairness requirements (if a property is not true, show a violating system run)? Which do become true if we assume weak fairness for all transitions? Which do become true if we assume strong fairness for all transitions? Explain your answers.

5. Consider the following version of the alternating bit protocol where messages are transmitted over an unreliable network that may loose or duplicate messages:

```
msgq = ackq = empty
Sender:
   sbit = sack = 0 // sent is arbitrary
   loop // produce message
     wait sack = sbit
     sent = any message
     sbit = 1-sbit
||
   loop // send (or resend) message
     sack != sbit -> send <sent,sbit> to msgq
||
   loop // receive acknowledgement
     receive ack from rcvq
Receiver:
   rbit = 0 // rcvd is arbitrary
   loop // receive message
     receive <m,b> from msgq
     if b != rbit then
       rcvd = m
       rbit = b
||
   loop // send acknowledgement
     send rbit to ackq
Network:
   loop
     remove entry from msgq
|| loop
     remove entry from ackq
```

Here both the sender and the receiver component operate on their own sets of local variables (sender variables *sbit*, *sack*, *sent* and receiver variables *rbit* and *rcvd*), communication between both components is achieved via two shared sequences (queues) *msgq* and *ackq*; sent messages are added to the end of a queue, received messages are removed from the

front of the queue. Each component has multiple threads that operate asynchronously with each other (indicated by the token $||$); also the network is modeled as a component by two threads that remove arbitrary entries from the message queues.

First model this system as a labeled transition system (which of above assignments can you combine into a single transition without affecting the correctness of the protocol?). Then also model this system as a "shared system" in the language presented in this book.

Formalize in LTL the properties:

a) Every received message was sent.

b) Every sent message will be received.

Which of these properties are true without fairness requirements (if a property is not true, show a violating system run)? Which do become true if we assume weak fairness for all transitions? Which do become true if we assume strong fairness for all transitions? Explain your answers. Please note that the answers are not necessarily the same as for Exercise 4.

Can this system be considered as a "refinement" of the system presented in Exercise 4? If yes, sketch an appropriate abstraction function and how the steps of the new system simulate the steps of the original one.

6. We consider a system which consists of a server and a set of clients. The server manages a pool of resources which clients request from the server by sending corresponding messages; the server grants these requests by sending corresponding replies. The central property of the system is that the server grants every resource to at most one client at a time, i.e., no two clients may simultaneously use the same resource.

   The problem becomes complex, because both requests and grants do not only refer to single resources; in particular, every client may request any set resources. However, the server may respond by a message that contains only some of the requested resources; if it does not immediately grant all resources, the server will send later further messages that grant more of them, until the complete request is satisfied. Furthermore, as soon as a client holds some of the requested resources, it may (even if its request has not yet been satisfied completely) return some of them to the server. However, a client may not send another request for new resources before it has received and returned all the resources from its previous request.

   To fairly satisfy requests from the various clients, the server keeps track of the sequence of still pending requests in the order in which they were received. The server grants a resource to a client if there is no earlier request from another client for the same resource.

   Formally model above system description as a labeled transition system. Then also describe this system as a shared system as well as a distributed system in the language of this book.

   Formulate in LTL the central property mentioned above as well as the property "Every request of every client will be eventually granted". Which of these properties are true without fairness requirements (if a property is not true, show a violating system run)? Which do become true if we assume weak fairness for all transitions? Which do become true if we assume strong fairness for all transitions? Explain your answers.

7. Consider the following system:

```
var x = 0
loop
  x = 1-x
```

   Model this system as a labeled transition system. Formulate the property "x is always zero or one" and prove it. Formulate the property "x eventually becomes one" and prove it.

8. Consider the following system:

```
var x = 0
loop
  x = x+1
```

   Model this system as a labeled transition system. Formulate the property "x is always a natural number" and prove it. Formulate the property "x eventually becomes every natural number n" and prove it (by induction on n).

9. Consider the following system:

```
var x = 0, y = 1, z = 0
loop          ||  loop
  x = 2*y     ||    y = x+1
```

   Model this system as a labeled transition system, formulate the property "y always is an odd number" as an LTL-formula, and prove this formula.

   Furthermore, formulate the property "y eventually becomes every odd number $2n + 1$" as an LTL-formula and prove this formula (by induction on n) under the assumption of "weak fairness" for the execution of each assignment.

10. Consider the following system:

```
var z = 0
loop                    ||  loop
  wait z = 0 -> z = 1   ||    wait z = 0 -> z = 1
  // critical region    ||    // critical region
  z = 0                 ||    z = 0
```

    Here `wait` $F \to x = T$ is the "guarded assignment statement" that is blocked unless F holds; its execution sets x to T.

    Model this system as a labeled transition system, formulate the property "no two processes are at the same time in the critical region" as an LTL-formula, and prove this formula. Furthermore, prove the formula "every process waiting to enter the critical region will eventually enter it" and prove it under the assumption of strong fairness for the execution of this statement.

11. Generalize the model of Exercise 10 to n processes and prove that the corresponding properties still hold.

12. Consider the following system:

```
var x = 0, y = 1, z = 0
loop                      ||  loop
  wait z = 0 -> z = 1     ||    wait z = 0 -> z = 1
  x = y                   ||    y = x+1
  x = 2*x                 ||    z = 1
  z = 0                   ||
```

Model this system as a labeled transition system, formulate the property "y always is an odd number" as an LTL-formula, and prove this formula. Furthermore, formulate the property "y eventually becomes every odd number n" as an LTL-formula and prove this formula.

13. Consider the alternating bit protocol of Exercise 4. Formulate the following property as an LTL-formula: "Always when the receiver is at the statement that copies *sent* into *rcvd*, the sender is at its `wait` statement" and prove it.

    Furthermore, prove that neither the sender nor the receiver are permanently blocked at their respective `wait` statements.

14. Prove that Peterson's Algorithm presented in Exercise 3 indeed guarantees the mutual exclusion property.

    Furthermore, prove that none of the two processes is forever blocked in the `while` loop (such that it eventually enters the critical region).