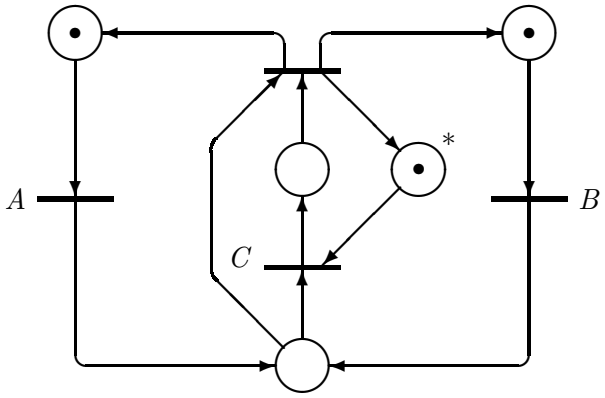
Solutions for CP Exercise Class 8

1. Solution for Concurrent Systems Exam December 2003, Problem 1

Question 1.1

(a)



[Note how C is prohibited in firing twice by the place $(*)$ corresponding to control of P_3]

(b) All three pairs (A, B), (B, C) and (A, C) can fire together and hence be executed concurrently. The latter two are easily seen by firing either A or B first.

(c) Disregarding the place $(*)$, the number of tokens on the remaining places is seen to be invariantly 2 (since any transition only *moves* tokens among these places). Hence A, B and C cannot execute concurrently.

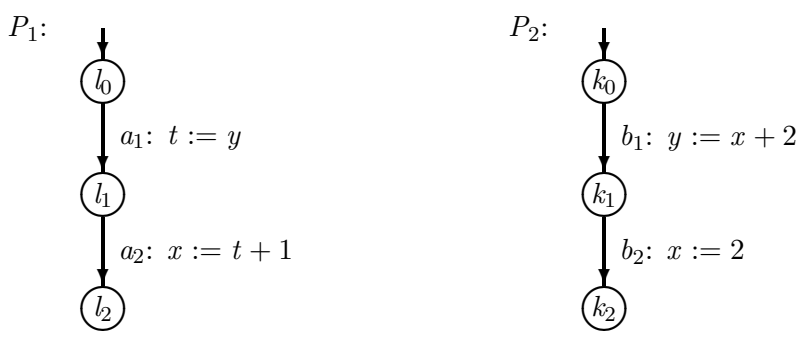
2. Solution for Concurrent Systems Exam December 2003, Problem 2

Question 2.1

The statements b, d, e, and f can be considered to be atomic since they only have only one critical reference each. Both a and c have two critical references.

Question 2.2

(a)



[Location and action labels not required.]

(b) Going through the 6 possible interleavings, the possible results for (x, y) are found to be:

$$(2,3), (2,2), (3,2), (1,2)$$

Question 2.3

P is preserved by a_2 and a_3. [By a_2 since $y < 0$ and P imply $x > 0$ and hence y also becomes positive.]

Q is preserved by all three actions. [Also by a_2 since it cannot be executed when Q holds.]

R is preserved only by a_2.

Question 2.4

(a) The sequence $(0,1)(1,2)$ repeated forever will satisfy all parts of F.

(b) Assuming F, only the guard of a_3 is constantly true and hence only a_3 is guaranteed to be eventually executed under weak fairness. [If $x \neq 0$ it must be positive due to $\Box x \geq 0$ and hence $y > x > 0$ imply $y > 1$.]

(c) Assuming F, the guards of a_1, a_2, and a_3 will be infinitely often true, and hence they will be eventually executed under strong fairness. [The guard of a_4 is not necessarily true, for instance, it is never true in the state sequence proposed in (a).]

3. Solution for Concurrent Systems Exam December 2003, Problem 3

Question 3.1

The first three calls of *put*() will enable one of the calls of *unload*() to succeed. The two remaing calls of *put*() will then both succeed leaving the server with $count = 2$. The second call of *unload*() will remain blocked waiting for acceptance by the server.

Question 3.2

```
monitor Batch

  var count : integer := 0;
      NonFull : condition;
      Full : condition;

  procedure unload() {
    while count < N do wait(Full);
    count := 0;
    signal(NonFull);
  }

  procedure put() {
    while count = N do wait(NonFull);
    count := count + 1;
    if count < N then signal(NonFull) else signal(Full);
  }

end
```