**2SD3 Assignment 3**

Rochan Muralitharan – muralr3

We will use these invariants to prove that this Coloured Petri Net is deadlock-free:

[inv1] m(p1) + m(p2) + m(p3) + m(p4) + m(p5) = ph1 + ph2 + ph3 + ph4 + ph5

[inv2] |m(p2)| + |m(p7)| = 4

[inv3] LF(m(p4)) + RF(m(p4)) + m(p6) = f1 + f2 + f3 + f4 + f5

Now we will consider these two cases:

1. m(p4) + m(p5) ≠ 0.

Then either return\_left\_fork or return\_right\_fork and exit\_dining\_room can be fired

1. m(p4) + m(p5) = 0.

Then from invariant [inv3] we will have:

LF(m(p3)) + m(p6) = f1 + f2 + f3 + f4 + f5

and from invariant [inv1]:

m(p1) + m(p2) + m(p3) = ph1 + ph2 + ph3 + ph4 + ph5

From the definition of LF(x) and the definition of RF(x), we know that we have x = ph1, ph2, ph3, ph4, ph5.

Thus if m(p3) ≠ 0 then the transition take\_right\_fork can be fired.

Also if m(p2) ≠ 0 then take\_left\_fork can be fired.

If m(p1) ≠ ph1 + ph2 + ph3 + ph4 + ph5, then either m(p3) ≠ 0 or m(p2) ≠ 0.

If m(p1) = ph1 + ph2 + ph3 + ph4 + ph5 then m(p2) = 0, and from invariant [inv2] | m(p7)| = 4, so enter\_dining\_room can be fired.

1. a)

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with low confidence

b)

c) In Java File Gas\_Station\_A3\_Q2.java

1. Graphical user interface, text, application, email

   Description automatically generatedA picture containing text, blackboard

   Description automatically generateda)

Graphical user interface, text, application, email

Description automatically generated3.c)

Here we can see that Meek.getcheese will clearly get starved, since Bold.getcheese will always be executed. Bold will always be given favour when there is a choice between meek and bold.

1. Graphical user interface, text, application

   Description automatically generated with medium confidencef
2. In Java File A3\_Q5.java
3. Graphical user interface, text, application, email

   Description automatically generatedD
4. a)
5. = : is equivalent to ( (p) r) p r. We have L(s0) = {r} so M, s0 . We also have L(s2) = {p,q} so M, s0
6. = EG r This statement translates to: There does not exist at least one path from all future states leading to r. We have r L(s0) and r L(s1) as L(s0) = {r} and L(s1) = {p,t,r}. There is an infinite path s0 s1  s1  s1  … , so M, s0 EG r and thus we can infer that M, s0 . Also, r L(s2) as L(s2) = {p, q} and thus M, s2 as the future includes the present.
7. = E( t U q) This statement translates to: There exists at least one path in where t will occur until q. As t L(s0) and t L(s2), we have M, s0 and M, s2 .
8. = F q This statement translates to: Some future state leads to q. As q L(s2) since L(s2) = {p,q}, and there are infinite paths s0 s­2 s­0  s­2 …, we have M, s0 . Also trivially, M, s2 as q L(s2) and the futures includes the present.

For the following questions we will assume the following:

“p precedes q” means that p must happen before q, and not at the same time

“p is followed by q” means that q must happen before p, and not at the same time

“p is between q and r” means that p does occur at the same time as q or r

b) “Event p precedes s and t on all computational paths”

Negation: “There exists a path where p does not precede s or does not precede t”

LTL: G(F p (p F s) (p F t))

CTL: AG(AF p AG(p AF s) AG(p AF t))

c) “Between the events q and r, p is never true but t is always true”

LTL: G(F p F r (q ( p U r) (q (F t U r)))

CTL: AG(AF q AF r) AG (q A( p U r))

d) “ is true infinitely often along every path starting at s”

LTL: s G(F )

CTL: s AG(AF )

e) “Whenever p is followed by q(after some finite amount of steps), then the system enters an ‘interval’ in which no r occurs until t”

LTL: G(p XG( q r U t))

CTL: AG(p AX AG( q A ( r U t)))

f) “Between the events q and r, p is never true”

LTL: G(F q F r (q ( p U r)))

CTL: AG(AF q AF r) AG(q A( p U r))

1. We will assume the following atomic predicates that characterize properties of processes:

lpri = local processing of reader i, i=1,2

lpwi = local processing of writer i, i=1,2

tri = reader i, i=1,2, requests reading

twi = writer i, i=1,2, requests writing

ri = reader i i=1,2, is reading

wi = writer i, i=1,2, is writing

To avoid any problems that might occur if we do not consider mutual exclusion, we will introduce some additional boolean variables(or atomic predicates):

turn = w1 (indicates the world where writer 1 will write)

turn = w2 (indicates the world where writer 2 will write)

turn = r (indicates the world where one or both readers will read)

Now the states can be identified by the atomic predicates of the form:

(sr1, sr2, sw1, sw2, turn)

Where:

sr1 {lpr1, tr1, r1} - status of reader 1

sr2 {lpr2, tr2, r2} – status of reader 2

sw1 {lpw1, tw1, w1} - status of writer 1

sw2 {lpw2, tw2, w2} – status of writer 2

turn {turn = w1, turn = w2, turn = r} – status of turns

Life of a reader follows the simple cycle:

(lpr1, \*, \*, \*, \*) (tr1, \*, \*, \*, \*) (r1, \*, \*, \*, \*) back to the beginning

Life of a writer follows a similar cycle:

(\*, \*, lpw1, \*, \*) (\*, \*, tw1, \*, \*) (\*, \*, w1, \*, \*) back to the beginning

However not all combinations of atomic predicates are allowed, for example:

sw1 = w1 sr1 r1 sr2 r2 sw2 w2

OR

sr1 = r1 sw1 w1 sw2 w2

Now we can establish safety and liveness properties in LTL and CTL:

Safety

LTL: G(w1 (w2 r1  r2))

CTL: AG(w1 (w2 r1  r2))

Liveness

LTL: G(tr1 F r1)

CTL: AG(tr1 AF r1)