Theory Assignment 3
CS5280

Darpan Gaur
CO21BTECH11004

Problem 4.1

$$s_1 = w_1(x)r_2(y)r_1(x)c_1r_2(x)w_2(y)c_2$$
$$s_2 = r_1(x)r_2(x)w_3(x)w_4(x)w_1(x)c_1w_2(x)c_2c_3c_4$$

2PL

s1

$$s_1 = wl_1(x)w_1(x)rl_2(y)r_2(y)r_1(x)wu_1(x)rl_2(x)r_2(x)c_1wl_2(x)wl_2(y)w_2(y)wu_2(y)ru_2(y)c_2$$

s2

We need to abort one of transactions t_1 or t_2. As there will be a deadlock condition, $rl_1(x)r1(x)rl_2(x)r2(x)$ after this point, t_1 and t_2 will be waiting for each other to release locks. So, we need to abort one of the transactions. Let's abort t_2. So, output for s_2 is:

$$s_2 = rl_1(x)r_1(x)rl_2(x)r_2(x)a_2wl_1(x)w_1(x)wu_1(x)ru_1(x)c_1$$
$$wl_3(x)w_3(x)wu_3(x)wl_4(x)w_4(x)wu_4(x)c_3c_4$$

O2PL

For $s1$ we can use the same schedule as 2PL. For $s2$, we need to abort t_1 and t_2, as t_3 and t_4 will share locks. So for operations $w_1(x)$ and $w_2(x)$, we need to wait untill t_3 and t_4 release locks. But t_3 and t_4 will be waiting for t_1 and t_2 to release locks. So, we need to abort t_1 and t_2. So, output for s_2 is:

$$s2 = rl_1(x)r_1(x)rl_2(x)r_2(x)wl_3(x)w_3(x)wl_4(x)w_4(x)a_1a_2wu_3(x)wu_4(x)c_3c_4$$

BTO

s1

Timestamp of t_1 is less than t_2. Also, we have only one conflicting operation $w_1(x)$ and $r_2(x)$ where $w_1(x) <_s r_2(x)$, and $ts(t_1) < ts(t_2)$. Hence, s_1 can be executed in BTO. Output schedule for s_1 is same as 2PL.

s2

Ordering of timestaps for t_1, t_2, t_3, t_4 is $t_1 < t_2 < t_3 < t_4$. Here, first $r_1(x), r_2(x), w_3(x)$ and $w_4(x)$ will be executed in order. Then for $w_1(x)$ we will check timestaps of t_1 and t_4 and since $ts(t_1) < ts(t_4)$, t_1 has to abort. Similarly, for $w_2(x)$ we will check timestaps of t_2 and t_4 and since $ts(t_2) < ts(t_4)$, t_2 has to abort. So ouput for s_2:

$$s_2 = r_1(x)r_2(x)w_3(x)w_4(x)a_1a_2c_3c_4$$

SGT

s1

There is only one conflicting pair $w_1(x)$ and $r_2(x)$, so only one edge in the SGT graph. Hence, no need to abort any transaction. So output for s_1 is same as 2PL.

s2

In figure 1, we can see that there are cycle bwtween nodes $(t_1, t_3), (t_1, t_4), (t_2, t_3)$ and (t_2, t_4). So t_1 and t_2 need to abort. So ouput for s_2 is:

$$s_2 = r_1(x)r_2(x)w_3(x)w_4(x)a_1a_2c_3c_4$$

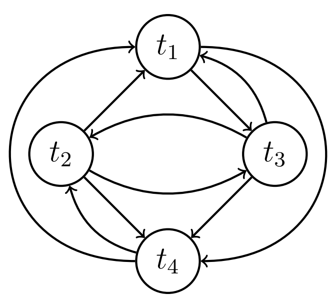

Figure 1: Serial (Conflict) Graph for s_2

Problem 4.9

In figure 2, t_1 will acquire lock on y and t_2 will acquire lock on x. As we can share locks in O2PL, t_2 will be able to acquire lock on y and t_1 will be able to acquire lock on x. But while releasing locks, we need to preserve order of locks. Now, t_1 will be waiting to release lock on x by t_2 and t_2 will be waiting to release lock on y by t_1. Hence, there will be a deadlock.

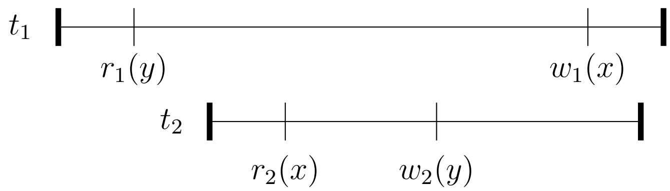

Figure 2: O2PL deadlock

Problem 4.12

The condition is semmingly natural but would lead to incorrect beahviour. If we remove a node, there is a possiblilty in future a cycle might be formed using the removed node, which will lead to incorrect behaviour of SGT protocol.

$$s = w_1(x)r_2(y)w_1(y)c_1r_3(z)w_2(z)c_2r_3(x)c_3$$

In figure 3, serialization graph is made using SGT protcol, and a cycle is formed. So, need to abort one of the transaction.
But if we use the given protocol in the question, we will remove t_1 after c_2 as t_1 is commited and active transaction at commit of t_1, i.e., t_2 is also commited. So, we will remove t_1 and the serialization graph will look like figure 4, which is acyclic (dashed line shows removed nodes). Hence suggested protocol is incorrect.

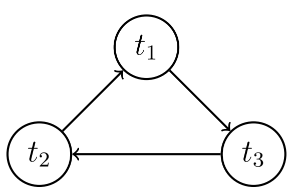

Figure 3: Without removing nodes

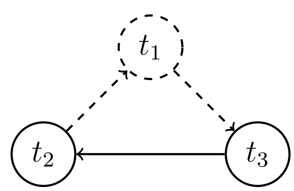

Figure 4: With removing nodes

Problem 4.16

$$s = r_1(x)r_2(x)r_1(y)r_3(x)w_1(x)w_1(y)c_1r_2(y)r_3(z)w_3(z)c_3r_2(z)c_2$$

BOCC

In this case, t_2 and t_3 will be aborted.

- t_2 abort: Here, val/write phase of t_1 is before t_2. $RS(t_2) \cap WS(t_1) \neq \phi$, as t_2 reads x and t_1 writes x. So, t_2 will be aborted.
- t_3 abort: Here, val/write phase of t_1 is before t_3. $RS(t_3) \cap WS(t_1) \neq \phi$, as t_3 reads x and t_1 writes x. So, t_3 will be aborted.

Final schedule will be:

$$s = r_1(x)r_2(x)r_1(y)r_3(x)w_1(x)w_1(y)c_1r_2(y)r_3(z)a_3r_2(z)a_2$$

FOCC

In this case t_1 will be aborted.

- t_1 abort: As $WS(t_1) \cap RS(t_2) \neq \phi$, as t_1 writes x and t_2 reads x. Also, $WS(t_1) \cap RS(t_3) \neq \phi$, as t_1 writes x and t_3 reads x. So, t_1 will be aborted.

Final schedule will be:

$$s = r_1(x)r_2(x)r_1(y)r_3(x)a_1r_2(y)r_3(z)w_3(z)c_3r_2(z)c_2$$