Assignment - 5

Rahul Bhansali 111401451

Jatin Garg 111462753

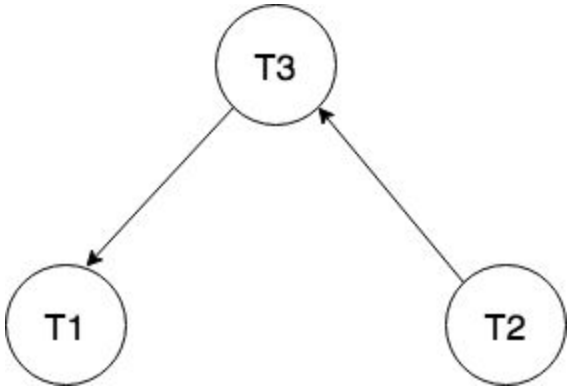
Exercise 18.2.5

Consider a schedule consisting of transactions as shown below:
T3 = w3a r3b
T1 = w1a
T2 = w2b

And schedule is S: **w3a w1a w2b r3b**
Now we will check our conditions:

(1) T1 precedes T2 in S. So first condition is satisfied.

(2) We will make a precedence graph for this schedule.



As there is no cycle, so condition 2 is also satisfied. So it is conflict serializable.

(3) Now the only serial schedule conflict equivalent to S is :

w2b w3a r3b w1a

And in this schedule T2 precedes T1.
So all 3 conditions are satisfied for this schedule.

Exercise 18.3.1

a) Example of a schedule that is prohibited by the locks:

r1(A) r2(B) w2(B) **r2(A)** w1(A) w2(A) r1(B) w1(B). Here, the problem is in the fourth step, where both transactions must hold a lock on A.

b) Legal Schedules:

Consider below table, where the column for A shows which transaction reads A first and which column for B shows which transaction reads B first. So, there are a total of 4 possible combinations.

A	B	#Legal Schedules
T1	T1	1 (Schedule is entire T1 followed by entire T2)
T2	T2	1 (Schedule is entire T2 followed by entire T1)
T1	T2	(4C2 X 4C2) = 36
T2	T1	0
Total		**38**

For the case of T1 reads A first and T2 reads B first: the read and write actions of T1 for A can be interleaved with the read and write actions for B of T2 => 4C2 ways. Similarly, the read and write actions for B of T1 can be interleaved with the read and write actions for A of T2 => 4C2 ways. Therefore, total combinations = 4C2 X 4C2 = 36.

The case of T2 reads A first and T1 reads B first is impossible, since if T2 reads A first => it must have already read B, and if T1 reads B first => it must have already read A. So both conditions cannot be satisfied and this is an impossible schedule to have.

c) Serializable schedules of the Legal Schedules:

A	B	#Serializable Schedules
T1	T1	1 (Schedule is entire T1 followed by entire T2)
T2	T2	1 (Schedule is entire T2 followed by entire T1)
T1	T2	(4C2 X 4C2) = 36

T2	T1	0
Total		**38**

The semantics for the schedule involves doing addition for A and multiplication for B. A schedule is serializable if the end result is the same as a serial schedule for both A and B. In this case, since addition and multiplication are commutative operators, it doesn't matter if actions on A happen from T1 or T2 first. Similarly for B. Hence, all legal schedules will be serializable.

d) Conflict serializable schedules

A	B	#Conflict-Serializable Schedules
T1	T1	1 (Schedule is entire T1 followed by entire T2)
T2	T2	1 (Schedule is entire T2 followed by entire T1)
T1	T2	0
T2	T1	0
Total		**2**

None of the 36 legal and serializable schedules of T1 reads A first and T2 reads B first will be conflict-serializable as the interleavings of reads and writes actions for either A or for B will not be "swappable" to generate the serialized schedule. One example of such a schedule:
r1(A) w1(A) **r2(B) w2(B) r1(B) w1(B)** r2(A) w2(A). Here read writes of T1 for B cannot be swapped ahead of read writes of T2 for B, as they are conflicting.

e) NO. For this case, the operations involve eventually adding 5 to A and multiplying B by 6. Since, addition and multiplication are both commutative, thus, it does not matter how the transaction actions are interleaved as eventually we will end up adding 5 to A and multiplying B with 6 i.e. the end result will be the same as a serializable schedule. And all legal schedules will be serializable.

18.3.4.)b.) r2(A) w2(A) w2(B)

Adding locks and unlocks in above transaction for each element accessed.

a.) l2(A) r2(A) w2(A) u2(A) l2(B) w2(B) u2(B)

We will first consider the total number of **consistent schedules**:

7! / (4! x 3!) = 35

As we can not change order for r2(A) and w2(A) , so 4 actions will be in fix order.

Now we will calculate number of **consistent non 2PL schedules.**
If any 1 lock/unlock seq occur before other lock/unlock sequence it will be non 2 PL. So there will be 2 such cases.

Total consistent non 2 PL cases = 2

Answer for part b of question is 2.

For part A we will subtract part two answer from total consistent schedules.
35 - 2 = 33

Answer for part a of question is 33.

Now we will calculate total 2PL schedules.

We have :
l2(A) l2(B) u2(A) u2(B)
These locks can be arranged in 2! Ways and unlocks can be arranged in 2! Ways.
So total is 4.
Now we have 3 actions left, r2(A) w2(A) r2(B).
Now in 2 PL we don't have to consider consistency or anything else. We just take into consideration that lock are taken before unlocks.
So there 3 actions can be arranged in lock/unlocks in 5 x 6 x 7 = 210 ways.

So total 2PL schedules are 210 x 4 = 840 ways.

Now will subtract consistent 2PL (part A of question) from 2PL to get inconsistent 2PL schedules.

840 - 33 = 807 ways

Answer for part c of question is 807.

Now to get neither consistent nor 2PL schedules we will subtract part a.), b.) and c.) answers from total schedules which are 7! = 5040.
Now 5040 - (2 + 33 + 807) = 4198 ways.

Answer for part d of question is 4198.

18.4.5.) Compatibility Matrices.

a.) Read, write and Multiplication by a constant locks.

	S	X	M
S	T	F	F
X	F	F	F
M	F	F	T

Shared lock is for read, exclusive lock is for write.

b.) Adding increment lock to above matrix.

	S	X	M	Inc
S	T	F	F	F
X	F	F	F	F
M	F	F	T	F
Inc	F	F	F	T

Exercise 18.4.7

If a cycle exists in the Precedence Graph of the schedule then the schedule will become non-conflict serializable. Now an edge already exists from T1 to T2 (r1(A) to w2(A)). We need to create an edge from T2 to T1.

a) Read
A cycle can be created if the missing action is **r2(c)**.

b) Write
A cycle will be created if the missing action is **w1(B)** or **w2(C)**.

c) Update
An update will consist of a read; update; write. Thus, a cycle will be created if the missing action is **Update1(B)** or **Update2(C).**

d) Increment
A cycle will be created if the missing action is **Inc1(B)** or **Inc2(C).**

18.6.3.) Adding increment locks to warning based protocol.

So we already have a compatibility matrix for shared, exclusive and intentional locks.

	IS	IX	S	X
IS	T	T	T	F
IX	T	T	F	F
S	T	F	T	F
X	F	F	F	F

Now we will add increment lock in this compatibility matrix. IInc is intentional increment lock.

	IS	IX	S	X	Inc	IInc
IS	T	T	T	F	F	T
IX	T	T	F	F	F	T
S	T	F	T	F	F	F
X	F	F	F	F	F	F
Inc	F	F	F	F	T	T
IInc	T	T	F	F	T	T

In this way we can add increment locks to a warning based protocol by using this lock compatibility matrix.