Name: Preethi chunduri- Assignment 2

14.5

Answer: Most of the concurrency control protocols (protocols for ensuring that only serializable schedules are generated) used in practice are based on conflict serializability—they permit only a subset of conflict serializable schedules. The general form of view serializability is very expensive to test, and only a very restricted form of it is used for concurrency control.

14.6

Answer: There is a serializable schedule corresponding to the precedence graph below, since the graph is acyclic. A possible schedule is obtained by doing a topological sort, that is, T1, T2, T3, T4, T5.

14.8

a. A schedule showing the Lost Update Anomaly:

T1	T2
Read(A)	
	Read (A)
	Write (A)
Write(A)	

In the above schedule, the value written by the transaction T2 is lost because of the write of the transaction T1.

b. Lost Update Anomaly in Read Committed Isolation Level

T1	T2
Lock-S(A)	
Read(A)	
Unlock(A)	
	Lock-X(A)
	Read(A)
	Write(A)
	Unlock(A)
	Commit
Lock-X(A)	
Write(A)	
Unlock(A)	
Commit	

The locking in the above schedule ensures the Read Committed isolation level. The value written by transaction T2 is lost due to T1's write.

14.12

Answer:

ACID properties are Atomicity, Consistency, Isolation and Durability

Atomicity: The atomicity property identifies that the transaction is atomic. An atomic transaction is either fully completed or is not begun at all.

Consistency: A transaction enforces consistency in the system state by ensuring that at the end of any transaction the system is in a valid state.

Isolation: When a transaction runs in isolation, it appears to be the only action that the system is carrying out at one time.

Durability: A transaction is durable in that once it has been successfully completed; all of the changes it made to the system are permanent.

14.14

Answer:

A schedule in which all the instructions belonging to one single transaction appear together is called a serial schedule. Serializable is used to find a non-serial type in existence, this process allows any transaction to execute concurrently without inferences and identifies if they are correct. Transaction execution has interleaving, meanwhile serial is always a serializable one.

14.15

Answer:

a. There are two possible executions: T13 T14 and T14 T13.
   Case 1:

Operations	A	B
Initially	0	0
After T13	0	1
After T14	0	1

Consistency met: A = 0 ∨ B = 0 ≡ T ∨ F = T

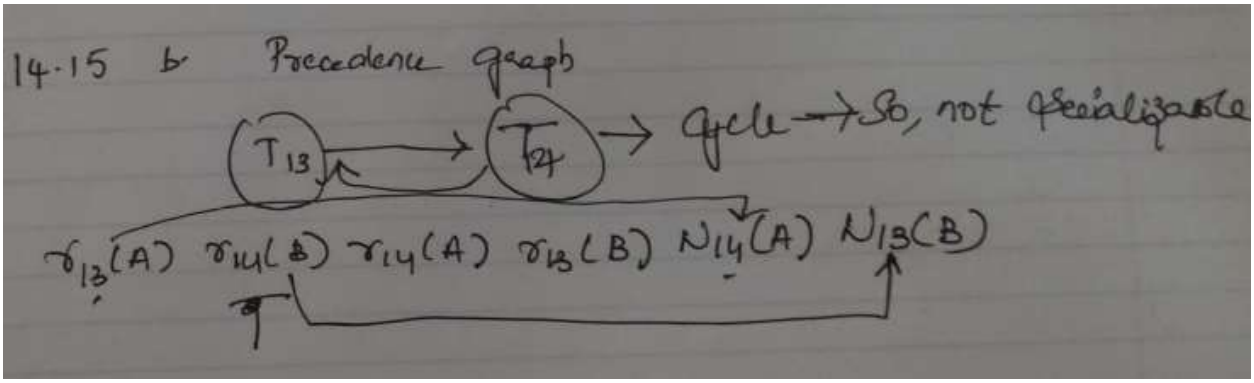
Case 2:

Operations	A	B
Initially	0	0
After T14	1	0
After T13	1	0

Consistency met: A = 0 ∨ B = 0 ≡ F ∨ T = T

b. Any interleaving of T1 and T2 results in a non-serializable schedule.

T_1	T_2
read(A)	
	read(B)
	read(A)
read(B)	
if $A = 0$ **then** $B = B + 1$	
	if $B = 0$ **then** $A = A + 1$
	write(A)
write(B)	



c. There is no parallel execution resulting in a serializable schedule. From part a. we know that a serializable schedule results in A = 0 ∨ B = 0. Suppose we start with T13 read(A). Then when the schedule ends, no matter when we run the steps of T14, B = 1. Now suppose we start executing T14 prior to completion of T13. Then T14 read(B) will give B a value of 0. So when T14 completes, A = 1. Thus B = 1 ∧ A = 1 → ¬ (A = 0 ∨ B = 0). Similarly for starting with T14 read(B).

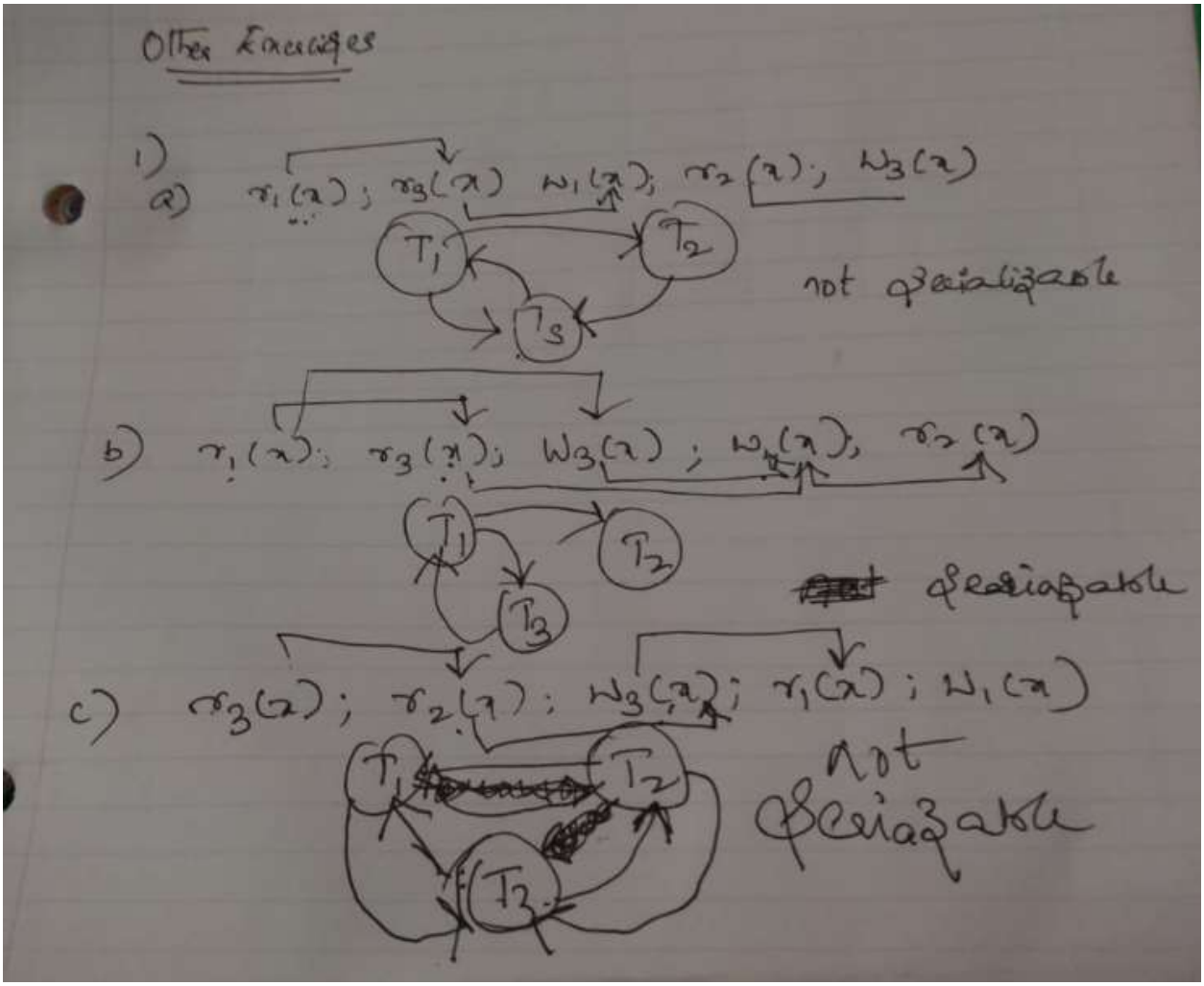
14.16

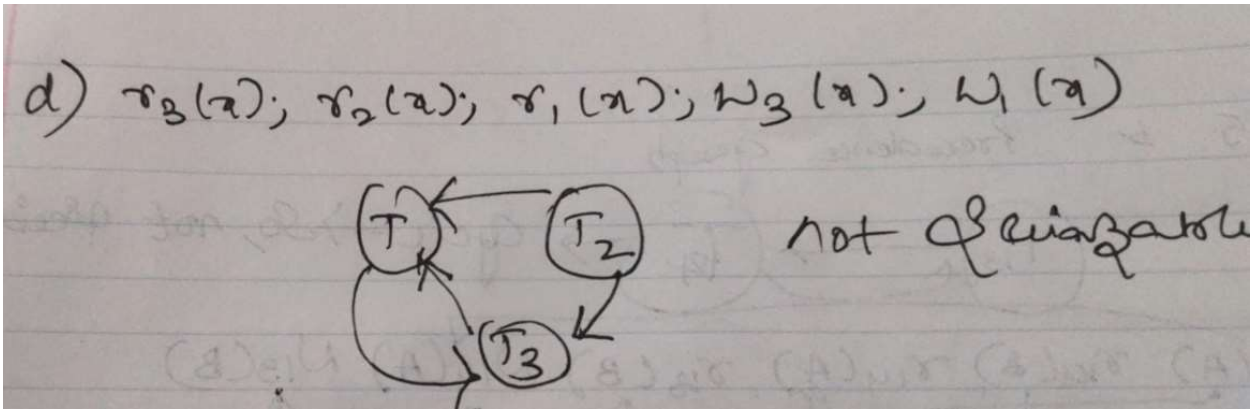
T1	T2
Read(A)	Read(B)
Unlock(A)	
	Write(B)
	Read(A)
	Write(A)
	Commit
Commit	

here the above schedule is serializable and T2 commits before T1. The unlock instruction is added to show how this schedule can occur even with strict two-phase locking, where exclusive locks are held to commit, but shared locks can be released early in two-phase manner.

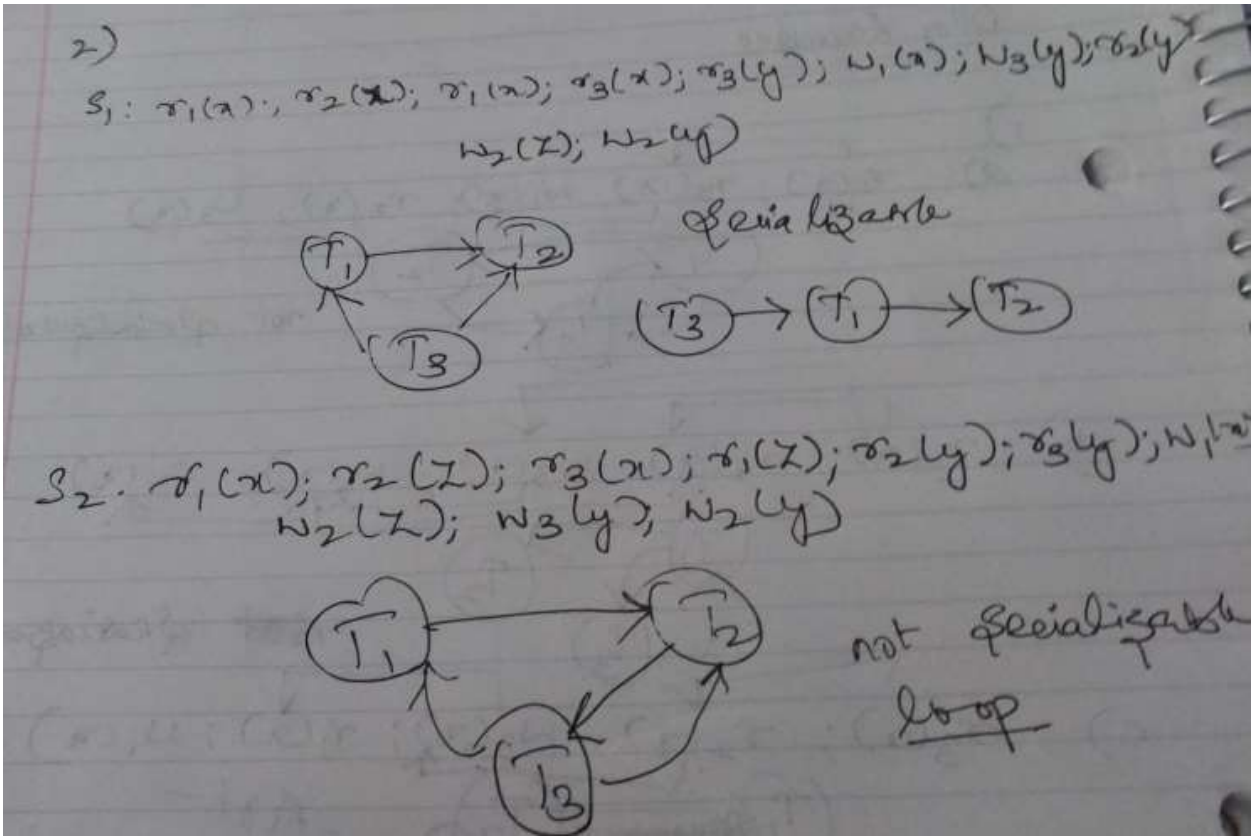
Other Exercises:

1.


Other Exercises
1)
a) $r_1(x)$; $r_3(x)$ $w_1(x)$; $r_2(x)$; $w_3(x)$
T_1
T_2
T_3
not serializable
b) $r_1(x)$; $r_3(x)$; $w_3(x)$; $w_2(x)$; $r_2(x)$
T_1
T_2
T_3
serializable
c) $r_3(x)$; $r_2(x)$; $w_3(x)$; $r_1(x)$; $w_1(x)$
T_1
T_2
T_3
not serializable


d) $r_3(x)$; $r_2(x)$; $r_1(x)$; $w_3(x)$; $w_1(x)$
T_1
T_2
T_3
not serializable

2.



3. A schedule is strict if it satisfies the following conditions:

Tj reads a data item X after Ti has written to X and Ti is terminated means aborted or committed.

Tj writes a data item X after Ti has written to X and Ti is terminated means aborted or committed.

S3 is not strict because in a strict schedule T3 must read X after C1 but here T3 reads X (r3(X))) before. Then T1 has written X (w1(X)) and T3 commits after T1.

S4 is not strict because in a strict schedule T3 must X after C1, but here T3 reads X (r3(X))) before T1 has written X (w1(X)) and T3 commits after T1.

Similarly with S5.

Cascadeless schedule:

Cascadeless schedule follows the below conditions:

Tj reads X only after Ti has written to X and terminated means aborted or committed.

S3 is not cascadeless schedule because T3 reads X(r3(X))before T1 commits;

S4 is not cascadeless schedule because T3 reads X(r3(X))before T1 commits;

S5 is not cascadeless schedule because T3 reads X(r3(X))before T1 commits;

But while comes to the definition of cascadeless schedules s3, s4 and s4 are not casacadeless and T3 is not affected if T1 is rolled back in any of the schedules, that is:

T3 does not have to roll back If T1 is rolled back. The problem occurs because these schedules are not serializable.

Recoverable Schedules:

Schedules that follows the below condition:

Tj commits after Ti if Tj has read any data item written by Ti.

So Schedule S3 is recoverable based on the above definition where rolling back to T1 doesnot effect T2 and T3.