CS3402 Tutorial 8:

1. Which of the following schedules is (conflict) serializable? For each serializable schedule, determine the equivalent serial schedules.

   (a) $r_1(X); r_3(X); w_1(X); r_2(X); w_3(X);$
   (b) $r_1(X); r_3(X); w_3(X); w_1(X); r_2(X);$
   (c) $r_3(X); r_2(X); w_3(X); r_1(X); w_1(X);$

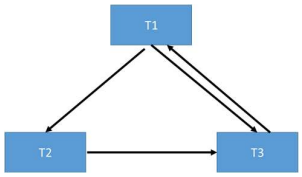
2. Consider the following concurrent schedule. Draw the serialization graph for the schedule. Is it conflict serializable?

Ta	Tb	Tc
	Read(x)	
Write(y)		
		Read(y)
	Write(y)	
Write(x)		
	Commit	
		Write(z)
Commit		
		Commit

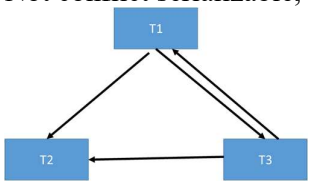
3. Consider schedules S_1, S_2 and S_3 below. Determine whether each schedule is strict, cascadeless, recoverable, or nonrecoverable. Determine the strictest recoverability condition that each schedule satisfies.
   (a) $r_1(X); w_1(X); r_2(X); r_1(Y); w_2(X); c_2; c_1;$
   (b) $r_1(X); w_1(X); r_2(X); r_1(Y); w_2(X); w_1(Y); c_1; c_2;$
   (c) $r_1(X); w_1(X); w_2(X); w_1(Y); c_1; r_2(X); c_2;$
   Can you change c) into a strict schedule?

CS3402 Tutorial 8:

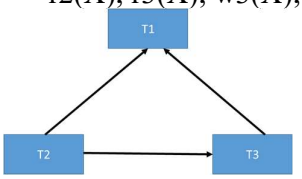
1. **Answer**:
   (a) Not conflict serializable, because the serialization graph contains cycle.



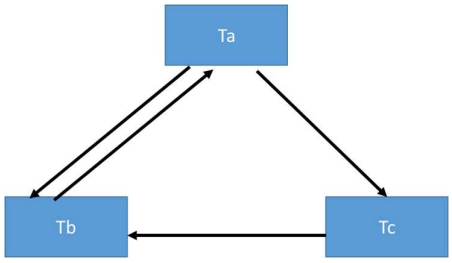
   (b) Not conflict serializable, because the serialization graph contains cycle.



   (c) Conflict serializable, because the serialization graph contains no cycle.
   it is equivalent to this serial schedule:
   r2(X); r3(X); w3(X); r1(X); w1(X); (or you can simply write T2, T3, T1)



2. **Answer**:



   It is not serializable as the schedule is cyclic.

3. **Answer**:
   (a) Non-recoverable, because T2 reads X written by T1， but T1 commits after T2.
   If T1 abort after T2 commits, the value of X is not recoverable.

   (b) Recoverable, because T2 reads X written by T1 and T2 commits after T1,
   which satisfy the condition of recoverable “ A schedule S is recoverable if no

transaction T in S commits until all transactions T′ that have written some item X that T reads have committed.”
It is not cascadeless， because T2 reads X written by T1 before T1 commits. If T1 fails, T1 has to be rolled back and T2 also need to be rolled back.

(c) Cascadeless. Because T2 reads X written by T1 after T1 commits. It satisfies the condition of Cascadeless schedule “Every transaction reads only the items that are written by committed transactions.”

It is not the strict schedule because T2 write X after T1 write X but before T1 commits. It does not satisfy the condition of strict schedule ： “A schedule in which a transaction can neither read or write an item X until the last transaction that wrote X has committed.”

Schedule (c) can be changed into the strict schedule:
r1(X); w1(X); w1(Y); c1; w2(X); r2(X); c2;