

Homework #5 (practice only - not graded)

1. Consider the three schedules below. For each one, answer the following questions and provide detailed explanation or proof for you answer:
- Is this schedule conflict-serializable?
 - Is this schedule recoverable if the transactions complete in the order specified?
 - Is this schedule cascadeless?
 - Is this schedule possible under a (non-strict) 2PL protocol?
 - Is this schedule possible under a strict 2PL protocol?
 - Is this schedule possible under a rigorous 2PL protocol?

T1	T2	T3
R(A)		
	R(B)	
W(C)		
		R(C)
	W(B)	
W(A)		
		R(B)
R(B)		
commit		W(C)
		commit
	abort	

Table 1: Schedule 1

T1	T2
	R(A)
R(B)	
R(A)	
W(A)	
	R(C)
W(B)	
commit	W(C)
	commit

Table 2: Schedule 2

T1	T2	T3
	R(A)	
R(B)		
		R(C)
R(A)		
		R(B)
W(A)		
	R(C)	
		W(C)
W(B)		
commit	W(C)	
	commit	R(A)
		commit

Assume this happens after commit of T1 and commit of T2, respectively.

Table 3: Schedule 3

2. Consider the following schedule on five transactions, where lock-X() and lock-S() denote requests for exclusive and shared locks, respectively. Is the schedule deadlocked? Prove your answer. If it is deadlocked discuss one method to recover. If it is not deadlocked, how would you avoid a deadlock?

T1	T2	T3	T4	T5
lock-S(A)				
		lock-S(C)		
	lock-X(B)			
			lock-X(D)	
				lock-S(E)
lock-S(E)				
		lock-S(G)		
	lock-S(A)			
				lock-X(C)
			lock-S(C)	
	lock-X(G)			
lock-S(D)				
				lock-S(B)
			lock-X(B)	
		lock-S(A)		

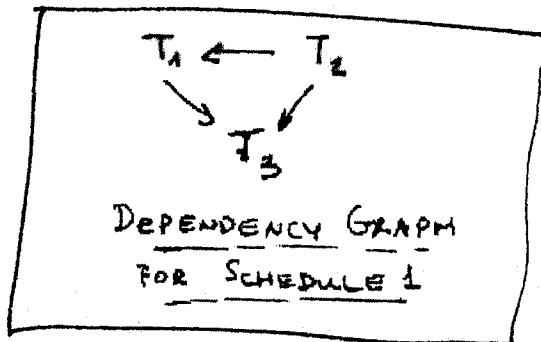
SAMPLE SOLUTIONS TO
HOMEWORK #5

CS 308
FALL 2005
POLYTECHNIC
UNIVERSITY

(1)

SCHEDULE 1

(a) YES. THE EQUIVALENT SERIAL SCHEDULE IS $T_2 \rightarrow T_1 \rightarrow T_3$ WHICH CAN BE ACHIEVED THROUGH SWAPPING. ALSO, THE DEPENDENCY GRAPH REVEALS NO CYCLES:



(b) No. T_3 HAS READ A VALUE OF 'B' THAT HAS BEEN WRITTEN BY T_2 . WHEN T_2 ABORTS, THIS VALUE MUST BE ROLLED BACK, BUT T_3 HAS ALREADY COMMITTED, SO THAT IS NOT POSSIBLE.

(c) No. BOTH T_1 AND T_3 HAVE READ A VALUE WRITTEN BY T_1 . FAILURE OF T_2 WILL CAUSE BOTH TO BE ROLLED BACK (AKA CASCADING ROLL BACK).

(d) No. T_3 WOULD NOT BE ABLE TO OBTAIN A LOCK FOR READING THE VALUE OF 'C', BECAUSE T_1 HAS ALREADY ACQUIRED A LOCK (EXCLUSIVE) ON IT, AND T_1 CANNOT

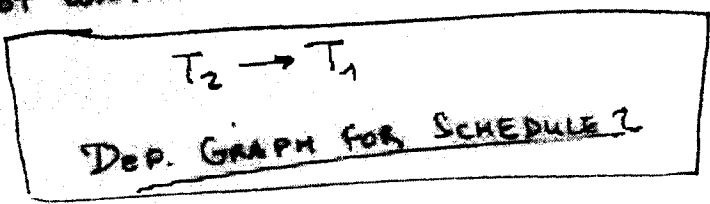
(e) No. THE SCHEDULE IS NOT TWO-PHASE (AND IT IS NOT CASCADELESS)

(f) No. SAME REASON

release any locks yet since it needs to acquire additional locks later.

SCHEDULE 2.

(a) YES. THE EQUIVALENT SERIAL SCHEDULE IS $T_2 \rightarrow T_1$. THE DEPENDENCY GRAPH DOES NOT CONTAIN CYCLES!



(b) YES. NEITHER OF THE TRANSACTION READS A VALUE WRITTEN PREVIOUSLY BY THE OTHER ONE.

(c) YES. ~~THERE ARE ONLY TWO TRANSACTIONS~~
~~CONCURRED~~ Same reason as (b).

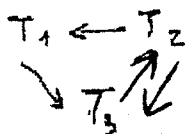
(d) No. T_1 CANNOT ACQUIRE AN EXCLUSIVE LOCK ON 'A' DUE TO T_2 ~~HAVING~~ ALREADY HAVING A SHARED LOCK ON 'A', WHICH IT CANNOT RELEASE YET SINCE IT NEEDS TO ACQUIRE ADDITIONAL LOCKS LATER.

(e) No. Due to (d).

(f) No. Due to (d)

SCHEDULE 3

(a) No. THE DEPENDENCY GRAPH REVEALS A CYCLE:



		<u>Edges:</u>
T_2 READS 'A' BEFORE	T_1 WRITES TO IT.	$T_1 \leftarrow T_2$
T_2 READS 'C' BEFORE	T_3 WRITES TO IT.	$T_2 \leftarrow T_3$
T_3 READS 'C' BEFORE	T_2 WRITES TO IT.	$T_3 \leftarrow T_2$
T_3 READS 'B' BEFORE	T_1 WRITES TO IT.	$T_3 \leftarrow T_1$

(b) YES. ONLY T_3 READS A VALUE^(OF 'A') WRITTEN BY ANOTHER TRANSACTION (T_1), AND T_1 COMMITS BEFORE T_3 .

~~(c) YES. IF T_1 OR T_2 IS ABORTED, ONLY T_3 WOULD BE ROLLED BACK. IF T_3 IS ABORTED, ONLY NO OTHER TRANSACTION WOULD BE ROLLED BACK.~~

YES, IF WE ASSUME THAT THE R(R) OF T_3 HAPPENS AFTER THE COMMIT OF T_2 . NO OTHERWISE

(d) No. T_1 CANNOT ACQUIRE AN EXCLUSIVE LOCK ON 'A' DUE TO T_2 'S SHARED LOCK ON 'A'.

(e) No, DUE TO (d).

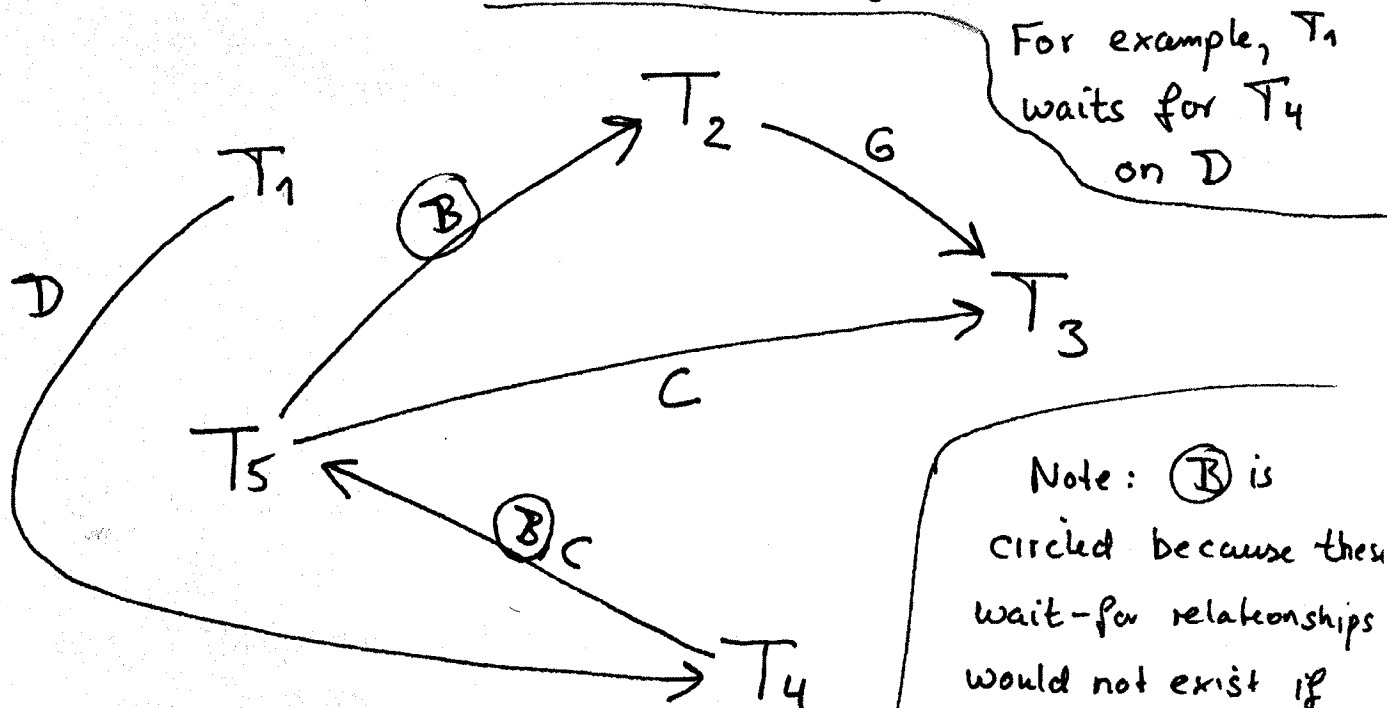
(f) No.

HW4, Problem 2:

Note: In the following we assume the use of 2PL.
This means that nobody can release any locks until they have acquired the last lock. The analysis also holds for strict and rigorous 2PL.

Wait-For graph:

For each wait-for edge, we show the object that is involved.



For example, T_1 waits for T_4 on D

Note: \textcircled{B} is circled because these wait-for relationships would not exist if T_2 and T_5 release their locks immediately after the last operation on the schedule shown

Thus, there is no cycle, and no deadlock. Transaction could finish in order T_3, T_2, T_5, T_4, T_1 .