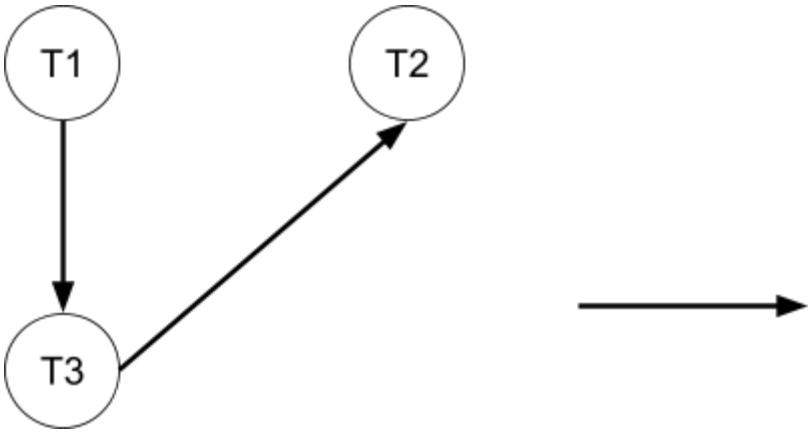
Problem Set 3: Part I

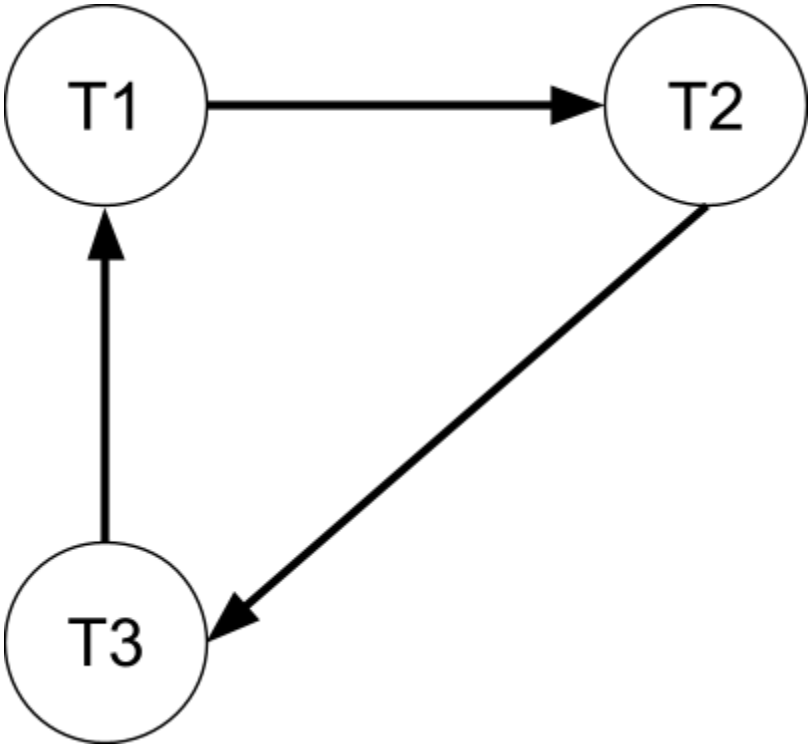
Problem 1: Conflict serializability

schedule 1
precedence graph:



conflict serializable (yes or no)? yes
equivalent serial schedule or explanation: There are no cycles => conflict serializable. Equivalent to T1; T3; T2.

schedule 2
precedence graph:



conflict serializable (yes or no)? No
equivalent serial schedule or explanation:

- cycle: T1 → T2 → T3 → T1 => not conflict serializable.

Problem 2: Two-phase locking and isolation

part	*yes or no?*	*explanation*
2.1	no	Once it begins releasing locks, it can't acquire any additional locks. In this partial schedule, u(A) must go after sl(B) and xl(C).
2.2	no	sl(B) only allows T2 to read B.
2.3	no	T1 doesn't hold all exclusive locks until it commits. In particular, T1 unlocks A before it commits. It happens the same to C in T1 and B and A in T2.
2.4	no	T2 performs a dirty read in this schedule (reading the value of A writes before T2 has commited), T2 commits before T1, which doesn't satisfy the requirement for a recoverable schedule.
2.5	no	because of the dirty read r(A) in T2.

Problem 3: Lock modes

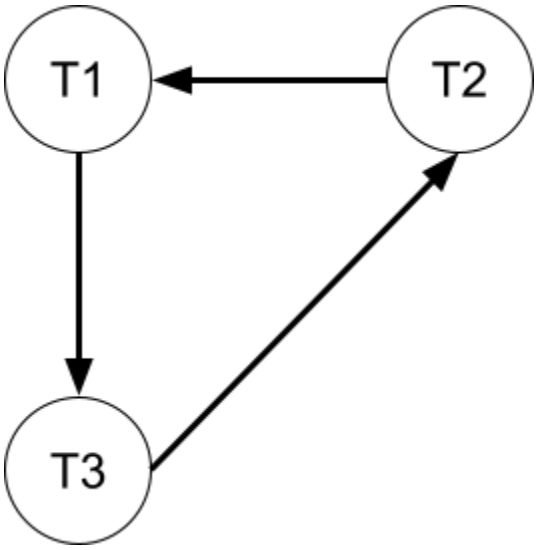
request	***granted or denied?***	***explanation***
sl4(A)	denied	the existing lock on A is an update lock.
sl3(B)	granted	there can be various shared locks on B.
ul5(A)	denied	there is an update lock on A in T1 => there can't be other update locks on A.
xl3(A)	granted	update locks can be upgraded if no other lock on item A.
ul4(C)	denied	the existing lock on C is an exclusive lock.
ul5(B)	granted	there are 2 shared locks on item B, T5 can still acquire an update lock for item B.

Problem 4: Deadlock detection

sequence 1: schedule

T1	T2	T3
		xl(D); w(D)
sl(D) denied; wait for T3		
	sl(E); r(E)	
	xl(F); w(F)	
		sl(F) denied; wait for T2
	xl(D) denied; wait for T1	

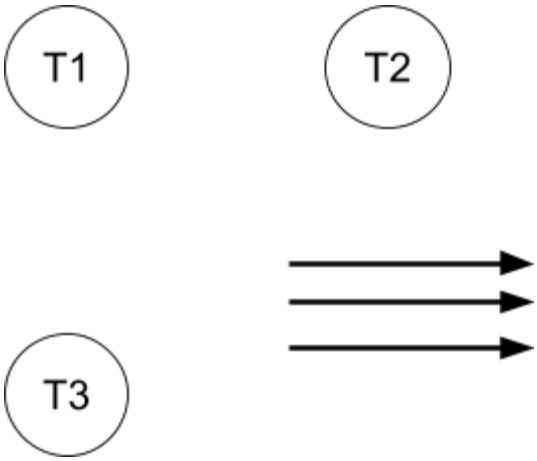
waits-for graph (if deadlock occurs):



sequence 2: schedule

T1	T2	T3
sl(X); r(X)		
xl(Y); w(Y)		
	sl(Y) wait for T1	
		sl(Z); r(Z)
		xl(X) wait for T1
w(X)		
c	r(Y)	
		w(X) c
	xl(Z); w(Z) c	

waits-for graph (if deadlock occurs):



Problem 5: Timestamps and multiple versions

5.1

request	**response of the system**	**any changes to state maintained for A *or* explanation of why action wasn't accepted**
w1(A)	allowed	WTS = 10
r3(A)	allowed	RTS = 30
w2(A)	denied; rolled back	TS = 20 < RTS = 30; now WTS = 0
w5(A)	allowed	WTS = 50
w3(A)	ignored	TS >= RTS but smaller than WTS = 50
r4(A)	denied; rolled back	TS = 40 < WTS = 50; RTS = 0

5.2

request	**response of the system**	**any changes to state maintained for A *or* explanation of why action wasn't accepted**
w1(A)	allowed	WTS = 10; c = false
r3(A)	denied; make wait	TS(T3) > WTS but c(A) = false
w2(A)	rolled back	TS(T2) < RTS(A); WTS(A) = 0
w5(A)	allowed	WTS = 50; c = false
w3(A)	denied; make wait	TS(T3) = RTS = 30; but TS(T3) < WTS
r4(A)	denied; rolled back	TS(T4) < WTS; RTS = 0
c1	allowed	c(A) = true
c2	doesn't happen	T2 has already been rolled back
c3	allowed	c(A) = true
r3(A)	now allowed!	RTS(A) = 30
w3(A)	now allowed!	WTS(A) = 30
c4	doesn't happen	there is no writer for T4
c5	allowed	c(A) = true

5.3

request	**response of the system**	**any changes to state maintained for A *or* explanation of why action wasn't accepted**
w1(A)	allowed	create A(10) with RTS = 0
r3(A)	allowed to read A(10)	RTS(A(10)) = 30
w2(A)	denied	TS(T2) < RST(A)
w5(A)	allowed	create A(50) with RTS = 0
w3(A)	allowed	create A(30) with RTS = 0
r4(A)	allowed to read A(50)	RTS(A(30)) = 40