Ejercicio A - Grafo de Precedencias

S1

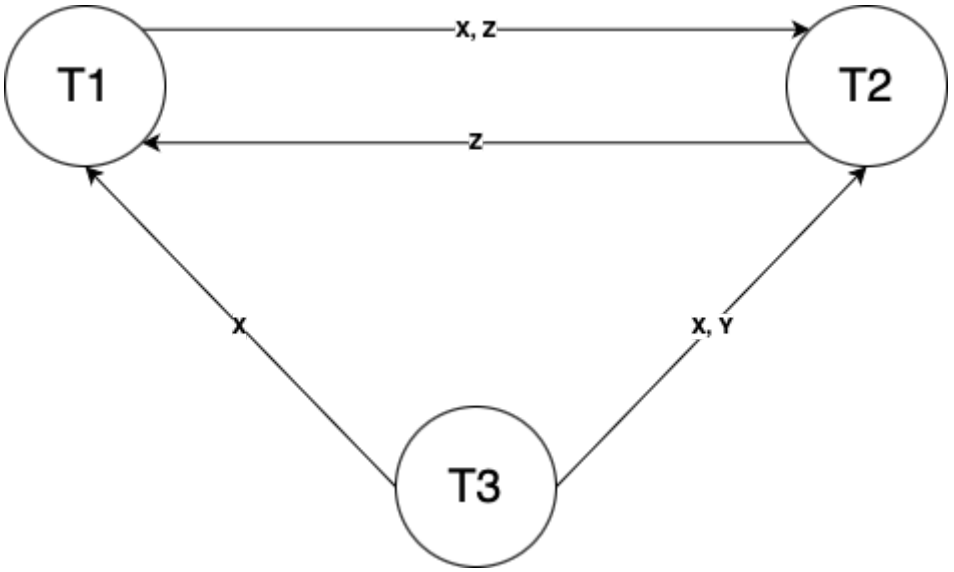
A partir de

S1: r1 (X); r2(Z); r1(Z); r3(X); w2(Z); w3(Y); w1(X); w3(Y); w1(Z); r2(Y); w2(X)

construimos la tabla de precedencias:

T1	T2	T3
R(X)		
	R(Z)	
R(Z)		
		R(X)
	W(Z)	
		W(Y)
W(X)		
		W(Y)
W(Z)		
	R(Y)	
	W(X)	

Traducimos en el grafo y obtenemos:



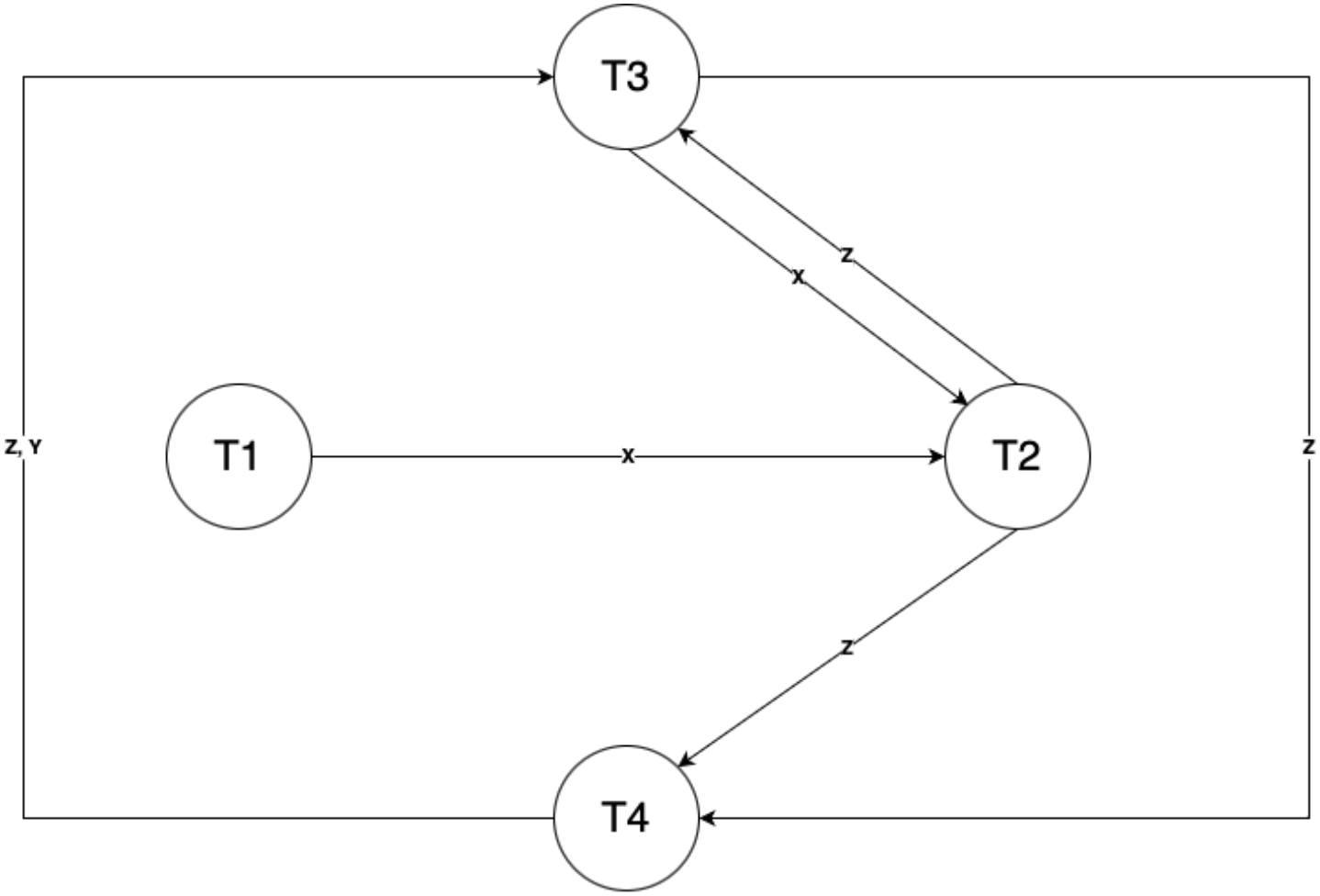
Como vemos que hay un bucle entre **T1** y **T2** por las operaciones que realizan con el recurso de *Z*, la planificación **no es serializable**.

S2

Nuevamente, construimos la tabla de precedencias y construimos el gráfico, esta vez para

S2: r1(X); r2(Z); r3(X); r4(Z); r2(X); r3(Z); w2(X); w4(Z); w3(Z); r4(Y); w4(Y); r3(Y);

T1	T2	T3	T4
R(X)			
	R(Z)		
		R(X)	
			R(Z)
	R(X)		
		R(Z)	
	W(X)		
			W(Z)
		W(Z)	
			R(Y)
			W(Y)
		R(Y)	



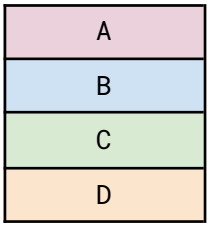
Nuevamente observamos que, por el bucle originado entre las transacciones **T3** y **T4** por la utilización del recurso *Z*, la transacción **no es serializable**.

Ejercicio B - Organización de Transacciones

B.1

T1=(r(A),r(B),w(A),w(B))
T2=(r(B),r(C),r(A),w(B))
T3=(r(A),r(C),w(C),w(A))
T4=(r(B),w(B),r(A),r(D),w(D))

Para ambos bloqueos, utilizaremos las siguientes referencias:



Bloqueo de dos fases básico

T1	T2	T3	T4	
		LOCK		Arbitrariamente, comienzo a ejecutar T3. Por protocolo básico, inmediatamente después de tomar el lock de cada uno de los recursos, acciono según lo que necesite.
		READ		
		LOCK		
		READ		
			LOCK	
			READ	
READ				Como los locks de todos los recursos que se quieren leer están tomados, aprovecho que la lectura es una funcionalidad compartida y ejecuto T1 y T2
READ				
	READ			
	READ			
	READ			
~~LOCK~~				Continúo con la ejecución de T1 que necesita hacer un lock de A, pero no puede ya que está tomado por T3. Se repite el procedimiento para T2 y ocurre lo mismo con el recurso tomado por T4.
	~~LOCK~~			
			WRITE	Continúo con la ejecución de T4 que al terminar de utilizar el recurso de B lo libera.
			UNLOCK	
			READ	
			LOCK	
			READ	
		WRITE		Continúo con la ejecución de T3, que como fueron liberados los recursos A y C, puede utilizarlos. Finalmente libera ambos pues no los vuelve a utilizar. Finaliza la ejecución de T3.
		UNLOCK		
		WRITE		
		UNLOCK		

	LOCK			
	WRITE			
	UNLOCK			Finaliza la ejecución de T2.
LOCK				
WRITE				
UNLOCK				
LOCK				
WRITE				
UNLOCK				Finaliza la ejecución de T1.
			WRITE	
			UNLOCK	Finaliza la ejecución de T4.

Bloqueo de dos fases conservador

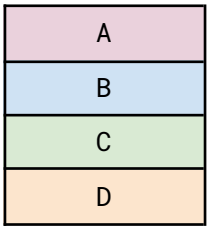
T1	T2	T3	T4	
			LOCK	Las transacciones 2 y 4 son las que más cantidad de recursos utilizan. Empiezo (arbitrariamente) por la 4. Por protocolo conservador, hago todos los locks que puedo.
			LOCK	
			LOCK	
~~LOCK~~				Comienzo la ejecución de T1. Por protocolo conservador, intento lockear todos los recursos que voy a utilizar. Como estos ya están tomados por T4, se me deniega el intento de lock y espero.
~~LOCK~~				
	~~LOCK~~			Comienzo la ejecución de T1. Por protocolo conservador, intento lockear todos los recursos que voy a utilizar. Como estos dos ya están tomados por T4, se me deniega todo el pedido (incluyendo C que no está tomado) de lock y espero.
	~~LOCK~~			
	LOCK			
		~~LOCK~~		
		LOCK		
			READ	
			WRITE	
			UNLOCK	
			READ	
			UNLOCK	
	LOCK			Comienzo con la ejecución de T2 y hago el pedido (nuevamente) de lock de todos los recursos que necesito. También tengo que pedir el lock de C pues en la solicitud anterior, todos los recursos se me fueron denegados.
	LOCK			
	LOCK			
~~LOCK~~				Nuevamente, por protocolo conservador, intento hacer el lock de todos los recursos que voy a utilizar y me los denegan pues están tomados por T2.
~~LOCK~~				
		~~LOCK~~		
		~~LOCK~~		
	READ			
	READ			

	UNLOCK			
	READ			
	UNLOCK			
	WRITE			
	UNLOCK			Finaliza la ejecución de T2.
			READ	
			WRITE	
			UNLOCK	Finaliza la ejecución de T4.
		LOCK		
		LOCK		
~~LOCK~~				
LOCK				
		READ		
		READ		
		WRITE		
		UNLOCK		
		WRITE		
		UNLOCK		Finaliza la ejecución de T3.
LOCK				
LOCK				
READ				
READ				
WRITE				
UNLOCK				
WRITE				
UNLOCK				Finaliza la ejecución de T1.

B.2

T1=(r(A),r(B),w(B),r(C),w(A))
T2=(r(D),r(B),r(C),w(B),w(D))
T3=(r(C),r(D),w(C),r(B))
T4=(r(A),w(A),r(D),w(D))

Para ambos bloqueos, utilizaremos las siguientes referencias:



Bloqueo de dos fases básico

T1	T2	T3	T4	
	LOCK			Arbitrariamente, comienzo la ejecución de T2 que toma el lock del recurso D.
	READ			
		LOCK		Continúo la ejecución con T3 que toma el lock del recurso C. Por último, lee el recurso D que es un recurso compartido por lo cual no necesita tomar el lock.
		READ		
		READ		
LOCK				
READ				
			READ	
			~~LOCK~~	Intento tomar el lock de A para escribir pero está tomado por T1, espero.
LOCK				
READ				
	READ			
WRITE				
UNLOCK				
			~~LOCK~~	Nuevamente intento tomar el lock de A y no puedo.
		WRITE		
		UNLOCK		
			~~LOCK~~	
LOCK				
READ				
UNLOCK				
	LOCK			
	READ			
	UNLOCK			

	LOCK			
		READ		Finalizo la ejecución de T3.
	WRITE			
	UNLOCK			
	WRITE			
	UNLOCK			Finalizo la ejecución de T2.
WRITE				Finalizo la ejecución de T1.
UNLOCK			LOCK	
			WRITE	
			UNLOCK	
			LOCK	
			READ	
			WRITE	
			UNLOCK	Finalizo la ejecución de T4.

Bloqueo de dos fases conservador

T1	T2	T3	T4	
	LOCK			De las 3 transacciones, elijo de forma arbitraria cualquiera de la que tenga que tomar la mayor cantidad de recursos. En este caso, elegimos T2 de T1, T2 y T3 (pues T4 usa sólo 2 recursos mientras que las demás utilizan 3). Por protocolo conservador, lockeo todos los recursos.
	LOCK			
	LOCK			
		~~LOCK~~		Para el resto de las transacciones disponibles, intento lockear todos los recursos que utiliza cada una. Si el lockeo de al menos un recurso es denegado, se deniega la totalidad del pedido y hay que volver a realizarlo en el futuro, cuando el CPU me de tiempo de procesamiento.
		~~LOCK~~		
		~~LOCK~~		
LOCK				
~~LOCK~~				
~~LOCK~~				
			LOCK	
			~~LOCK~~	
	READ			Ejecuto T2 hasta que libera alguno de los recursos previamente solicitados.
	READ			
	READ			
	UNLOCK			
	WRITE			
	UNLOCK			
LOCK				
LOCK				
LOCK				
		~~LOCK~~		

		~~LOCK~~		
		~~LOCK~~		
			~~LOCK~~	
			~~LOCK~~	
READ				
READ				
WRITE				
UNLOCK				
READ				
UNLOCK				
		LOCK		
		LOCK		
		~~LOCK~~		
			~~LOCK~~	
			~~LOCK~~	
	WRITE			
	UNLOCK			
		LOCK		Finaliza la ejecución de T2.
		LOCK		
		LOCK		
			~~LOCK~~	
			~~LOCK~~	
WRITE				
UNLOCK				Finaliza la ejecución de T1.
			~~LOCK~~	
			~~LOCK~~	
		READ		
		READ		
		UNLOCK		
		WRITE		
		UNLOCK		
			LOCK	
			LOCK	
			READ	
			WRITE	
			UNLOCK	
			READ	
		READ		

		UNLOCK		Finaliza la ejecución de T3.
			WRITE	
			UNLOCK	Finaliza la ejecución de T4.