Precedence Graph & Wait For Graph Tutorial

By: Virendra Kumar Meghwal

Precedence Graph (PG)

- Precedence Graph or Serialization graph is used commonly to test Conflict Serializability (CS) of a schedule.
- It is a directed graph (V, E).
  - where $V = \{T_1, T_2, T_3 \ldots T_n\}$ is a set of nodes (transactions) and
  - $E = \{E_1, E_2, E_3 \ldots E_m\}$ is a set of edges.
- Graph contains one node for each Transaction T_i .
- You can optionally label the edge by the name of the data item(s) which are used to create the edge.

Algorithm for C.S. using PG

1. For each transaction T_i participating in schedule S, create a node labeled T_i in the precedence graph.
2. For each case in S where T_j executes a read_item(X) after T_i executes a write_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
3. For each case in S where T_j executes a write_item(X) after T_i executes a read_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
4. For each case in S where T_j executes a write_item(X) after T_i executes a write_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
5. The schedule S is serializable if and only if the precedence graph has nocycles.

Understanding PG

- There are 3 operations that cause conflict:
  1. R-W [or R(x) – W(x)]
  2. W-R [or W(x) – R(x)]
  3. W-W [or W(x) – W(x)]
- No conflict is caused by R-R operation [R(x) – R(x)]
- Conflict can occur only if operations occur in **different transactions**.

Example -1

- Consider the following Schedule S:

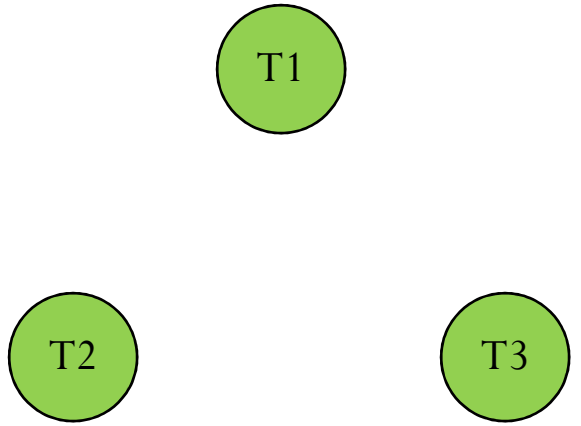
T1	T2	T3
R(x)		
		R(y)
		R(x)
	R(y)	
	R(z)	
		W(y)
	W(z)	
R(z)		
W(x)		
W(z)		

Example -1

- S:

T1	T2	T3
R(x)		
		R(y)
		R(x)
	R(y)	
	R(z)	
		W(y)
	W(z)	
R(z)		
W(x)		
W(z)		

- As our schedule S is having 3 transactions, we will draw 3 nodes.

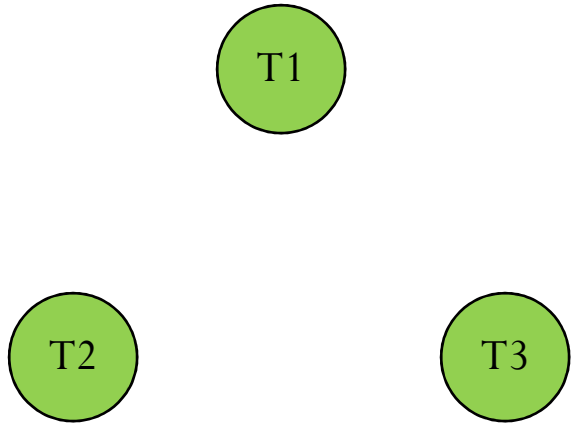


Example -1

- S:

Step	T1	T2	T3
1	**R(x)**		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Now consider each step one by one and look for conflicting operation in the remaining transactions in downward direction. (e.g. for step 1 look for conflict in T2 & T3)
- No conflict for step 1.

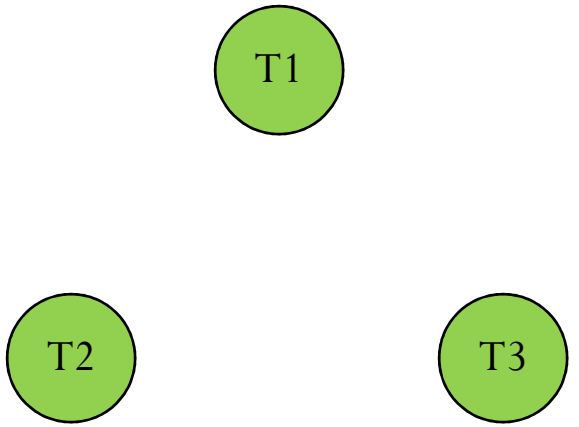


Example -1

- S:

Step	T1	T2	T3
1	R(x)		
2			**R(y)**
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Now repeat for step 2.
- No conflict for step-2.



Example -1

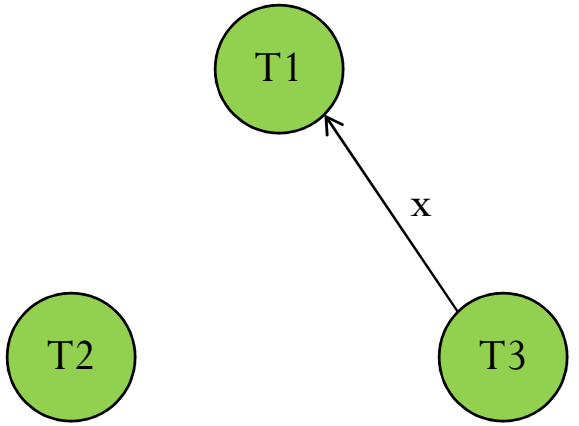
- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Now repeat for step 3.
- Step 3 of T3 conflict with step 9 of T1 .

1. $R_3(x) \rightarrow W_1(x)$ conflicting so draw edge T3 -> T1

   [**Meaning of above statement**: R(x) of T3 is conflicting with W(x) of T1. Thus, we can say that T3 transaction should be completed before executing transaction T1.]



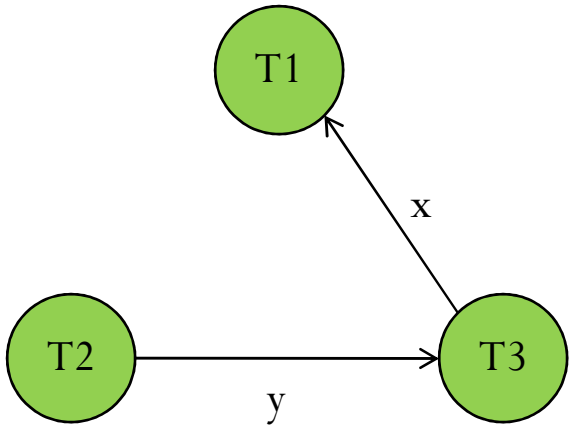
Example -1

- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Now repeat for step 4.
- Step 4 of T2 conflict with step 6 of T3 .

1. $R_2(y) \rightarrow W_3(y)$ conflicting so draw edge T2 -> T3



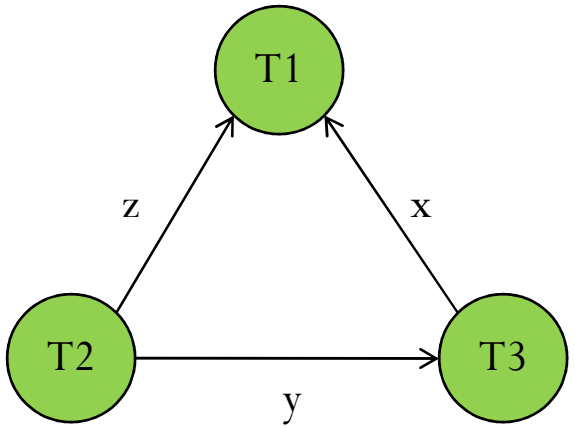
Example -1

- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Now repeat for step 5.
- Step 5 of T2 conflict with step 10 of T1 .

1. $R_2(z)$ -> $W_1(z)$ conflicting so draw edge T2 -> T1

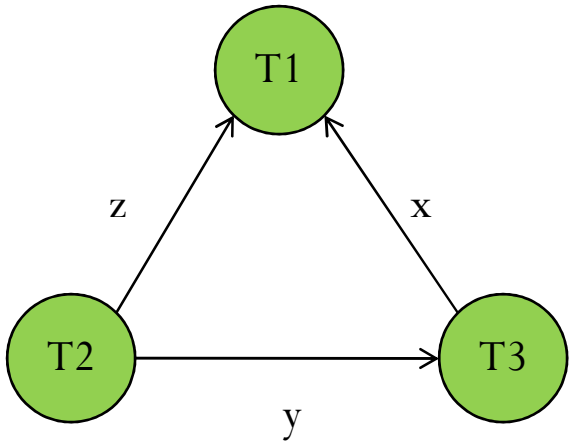


Example -1

- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Now repeat for step 6.
- No conflict.

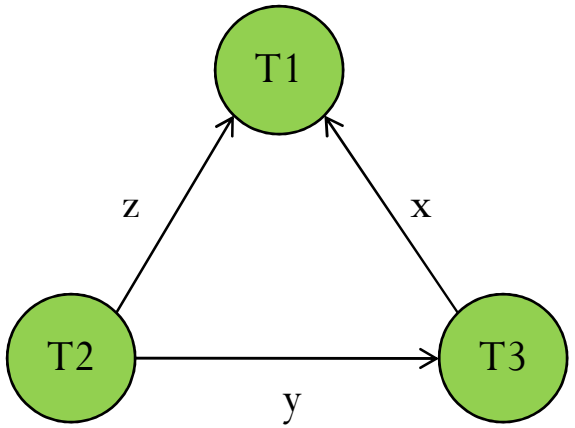


Example -1

- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Now repeat for step 7.
- Step 7 of T2 conflict with step 10 of T1 .
  1. $R_2(z) \rightarrow W_1(z)$ conflicting so draw edge T2 -> T1. But this edge already exists, so no need to redraw.

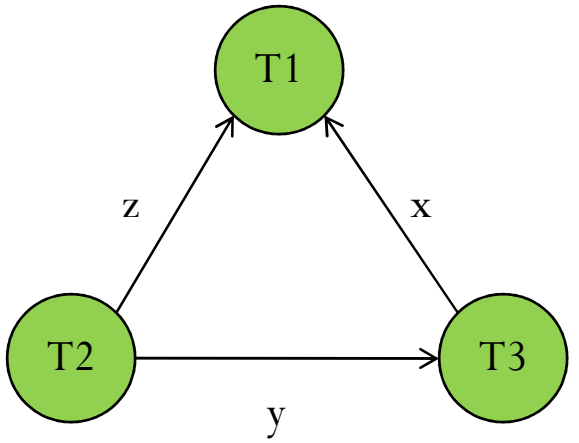


Example -1

- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Now repeat for step 8, 9 & 10.
- No conflict.
- So, our final graph is as follows:

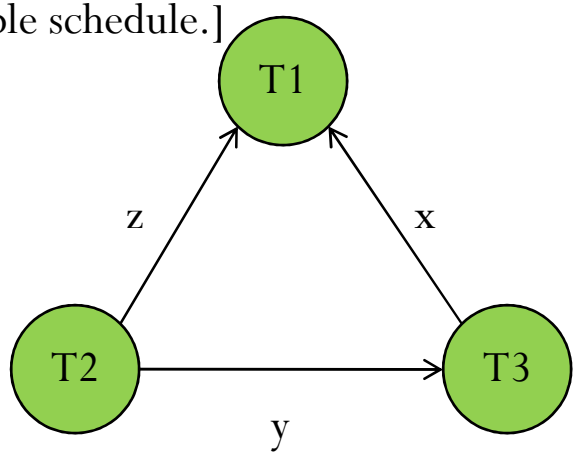


Example -1

- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Since there are no cycles, so there this schedule is conflict serializable.
- That is it is a serializable schedule.
- So, we can create a serial schedule for S.
- [**Note:** Writing name of the data items on the edges is optional. Because it does not matter if a cycle is formed by same data items or different data items. A cycle simply means the schedule is non conflict serializable schedule.]

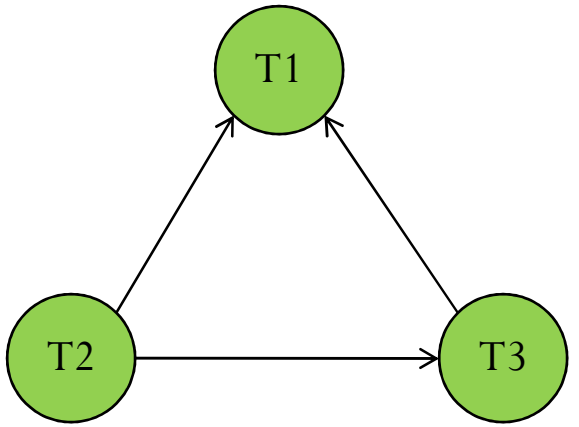


Example -1 (Find Serial Schedule)

- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Find the node with In-degree = 0 (means no incoming edges).
- Delete that node and its respective edges.
- Here, T2 is having in-degree =0

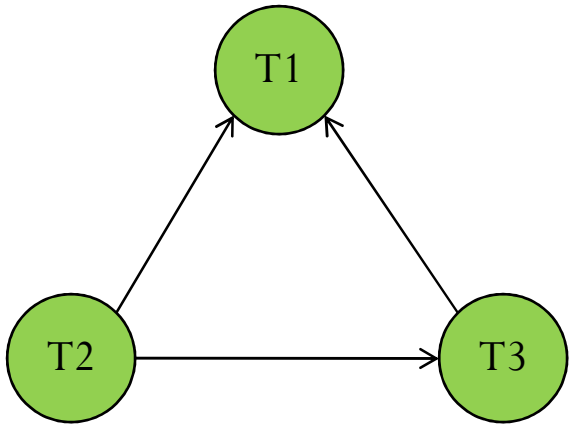


Example -1 (Find Serial Schedule)

- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Remove T2 and its corresponding edges.
- Serial Schedule = T2 ->

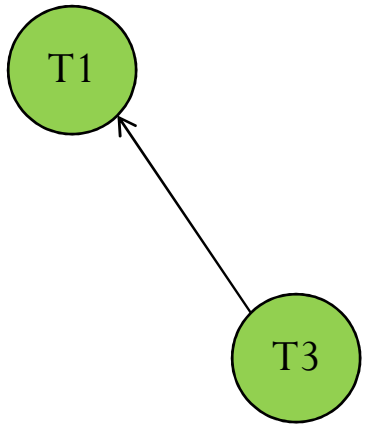


Example -1 (Find Serial Schedule)

- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Remove T2 and its corresponding edges.
- T2 becomes the first transaction of serial schedule
- Serial Schedule = T2 ->

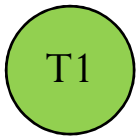


Example -1 (Find Serial Schedule)

- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Now T3 is having in-degree = 0.
- So delete T3.
- T3 becomes 2nd transaction of serial schedule.
- Serial Schedule = T2 -> T3 ->



Example -1 (Find Serial Schedule)

- S:

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

- Only one node remaining.
- Thus, T1 is the last transaction.
- Serial Schedule = T2 -> T3 -> T1

Example -1 (Find Serial Schedule)

- Non Serial Schedule S.

Step	T1	T2	T3
1	R(x)		
2			R(y)
3			R(x)
4		R(y)	
5		R(z)	
6			W(y)
7		W(z)	
8	R(z)		
9	W(x)		
10	W(z)		

=

- Serial Schedule = T2 -> T3 -> T1

Step	T1	T2	T3
1		R(y)	
2		R(z)	
3		W(z)	
4			R(y)
5			R(x)
6			W(y)
7	R(x)		
8	R(z)		
9	W(x)		
10	W(z)		

Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(x)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Find whether the given schedule is conflict serializable or not. If it is conflict serializable find all possible serial schedules.

- **Trick:** ignore **commit** operation and then proceed as in previous example.

Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Lets first find all conflicting pairs of operations.

1. $R_2(x) \rightarrow W_3(x)$ → T2 → T3
2. $R_2(x) \rightarrow W_1(x)$ → T2 → T1
3. $W_3(x) \rightarrow W_1(x)$ → T3 → T1
4. $W_3(x) \rightarrow R_4(x)$ → T3 → T4
5. $W_1(x) \rightarrow R_4(x)$ → T1 → T4
6. $W_2(y) \rightarrow R_4(y)$ → T2 → T4

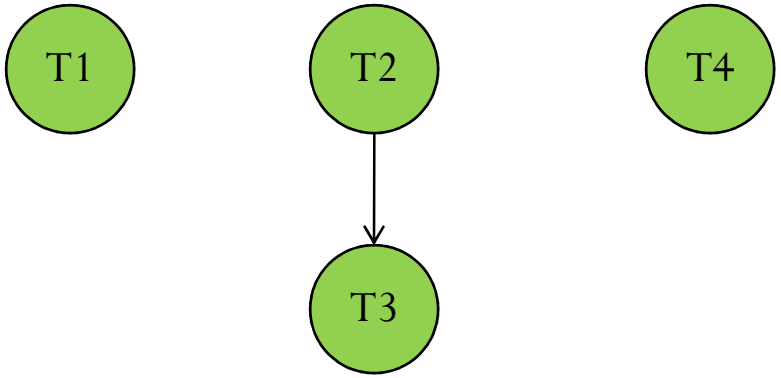
Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Lets first find all conflicting pairs of operations.

1. $R_2(x) \rightarrow W_3(x)$ → **T2 → T3**
2. $R_2(x) \rightarrow W_1(x)$ → T2 → T1
3. $W_3(x) \rightarrow W_1(x)$ → T3 → T1
4. $W_3(x) \rightarrow R_4(x)$ → T3 → T4
5. $W_1(x) \rightarrow R_4(x)$ → T1 → T4
6. $W_2(y) \rightarrow R_4(y)$ → T2 → T4



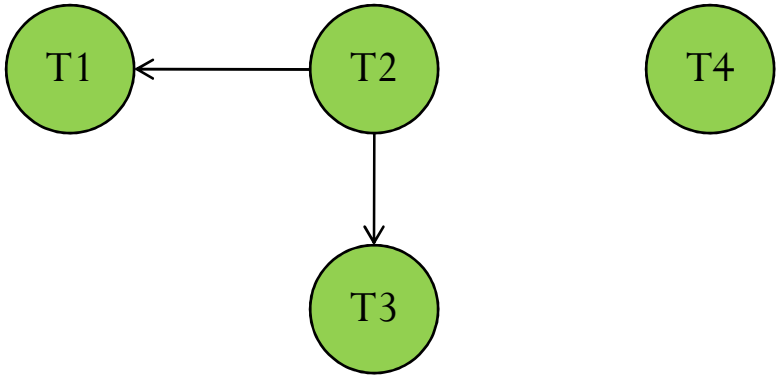
Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Lets first find all conflicting pairs of operations.

1. $R_2(x) \rightarrow W_3(x)$ → T2 → T3
2. $R_2(x) \rightarrow W_1(x)$ → **T2 → T1**
3. $W_3(x) \rightarrow W_1(x)$ → T3 → T1
4. $W_3(x) \rightarrow R_4(x)$ → T3 → T4
5. $W_1(x) \rightarrow R_4(x)$ → T1 → T4
6. $W_2(y) \rightarrow R_4(y)$ → T2 → T4



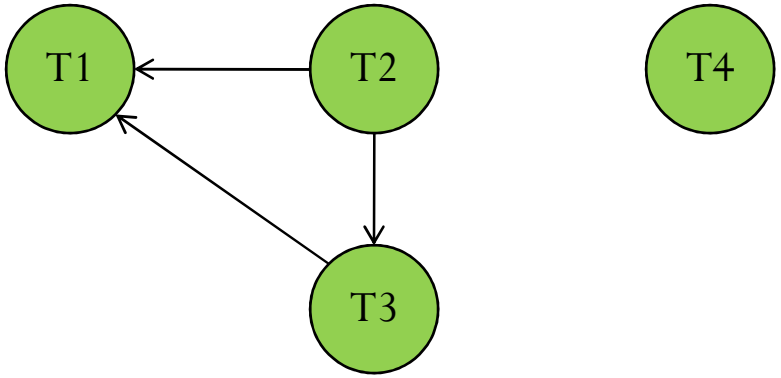
Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Lets first find all conflicting pairs of operations.

1. $R_2(x) \rightarrow W_3(x)$ → T2 → T3
2. $R_2(x) \rightarrow W_1(x)$ → T2 → T1
3. $W_3(x) \rightarrow W_1(x)$ → **T3 → T1**
4. $W_3(x) \rightarrow R_4(x)$ → T3 → T4
5. $W_1(x) \rightarrow R_4(x)$ → T1 → T4
6. $W_2(y) \rightarrow R_4(y)$ → T2 → T4



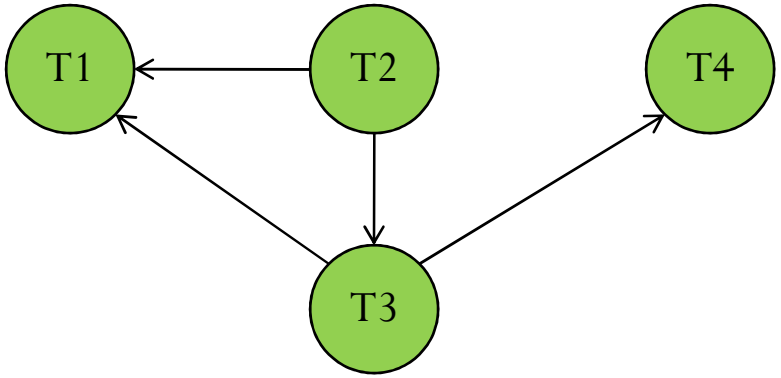
Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Lets first find all conflicting pairs of operations.

1. $R_2(x) \rightarrow W_3(x)$ → T2 → T3
2. $R_2(x) \rightarrow W_1(x)$ → T2 → T1
3. $W_3(x) \rightarrow W_1(x)$ → T3 → T1
4. $W_3(x) \rightarrow R_4(x)$ → **T3 → T4**
5. $W_1(x) \rightarrow R_4(x)$ → T1 → T4
6. $W_2(y) \rightarrow R_4(y)$ → T2 → T4



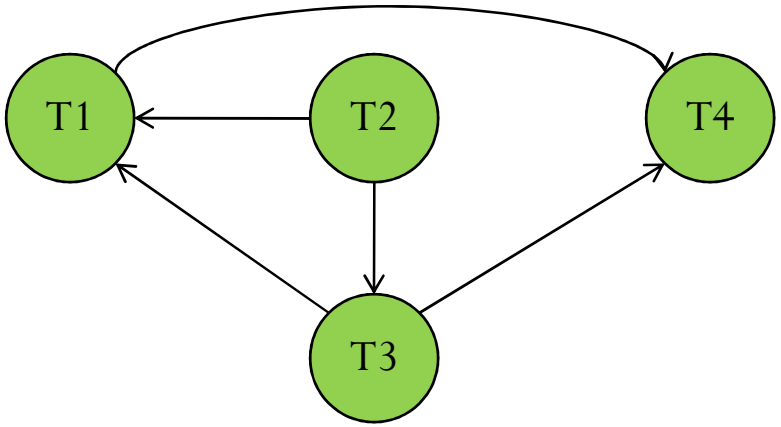
Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Lets first find all conflicting pairs of operations.

1. $R_2(x) \rightarrow W_3(x)$ → T2 → T3
2. $R_2(x) \rightarrow W_1(x)$ → T2 → T1
3. $W_3(x) \rightarrow W_1(x)$ → T3 → T1
4. $W_3(x) \rightarrow R_4(x)$ → T3 → T4
5. $W_1(x) \rightarrow R_4(x)$ → **T1 → T4**
6. $W_2(y) \rightarrow R_4(y)$ → T2 → T4



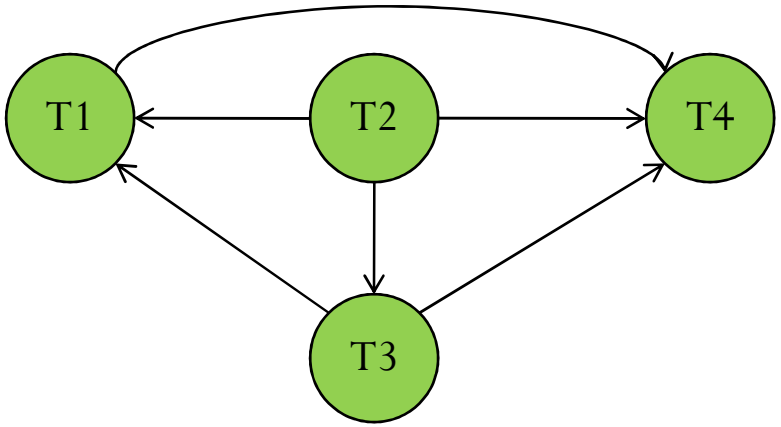
Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Lets first find all conflicting pairs of operations.

1. $R_2(x) \rightarrow W_3(x)$ → T2 → T3
2. $R_2(x) \rightarrow W_1(x)$ → T2 → T1
3. $W_3(x) \rightarrow W_1(x)$ → T3 → T1
4. $W_3(x) \rightarrow R_4(x)$ → T3 → T4
5. $W_1(x) \rightarrow R_4(x)$ → T1 → T4
6. $W_2(y) \rightarrow R_4(y)$ → **T2 → T4**

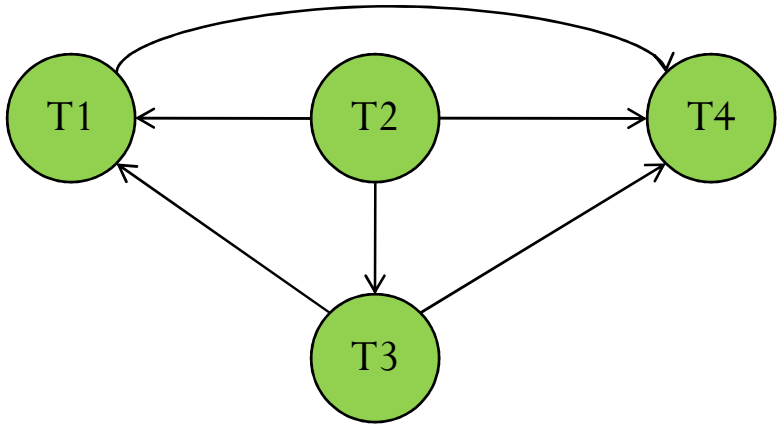


Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Clearly, there is no cycle.
- Thus the given schedule S is conflict serializable.
- Thus, a serial schedule is possible which will provide us the same results.
- Lets find the serial schedules.
- Lets look for node with in-degree = 0.
- Clearly, node T2 has in-degree = 0.
- So, we will remove this node and include T2 in serial schedule.

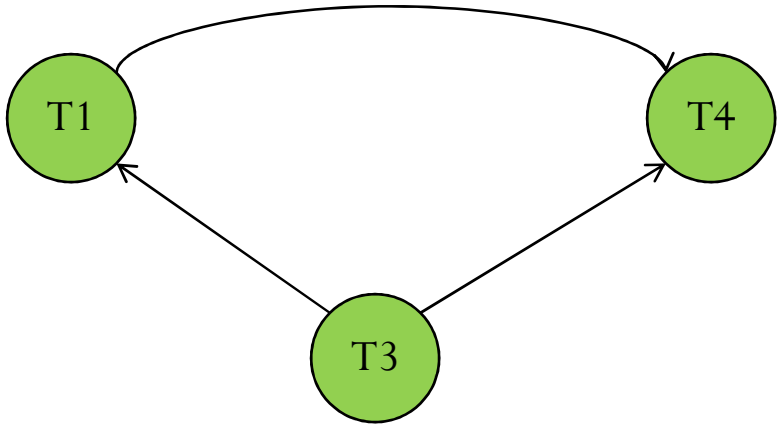


Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Serial Schedule: T2 →
- Now , node T3 has in-degree = 0
- So include this in serial schedule and remove from graph.

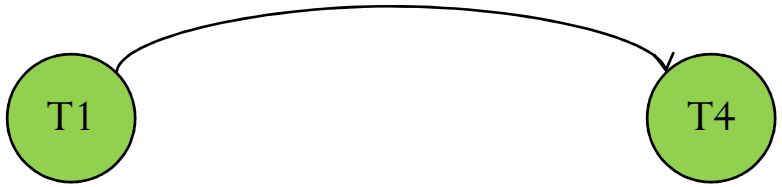


Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Serial Schedule: T2 → T3 →
- Now , node T1 has in-degree = 0
- So include this in serial schedule and remove from graph.



Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Serial Schedule: T2 → T3 → T1 →
- Now , node T4 has in-degree = 0
- So include this in serial schedule and remove from graph.

Example -2

- Non Serial Schedule S.

Step	T1	T2	T3	T4
1		R(x)		
2			W(x)	
3			commit	
4	W(x)			
5	commit			
6		W(y)		
7		R(z)		
8		commit		
9				R(x)
10				R(y)
11				commit

- Thus we have our serial schedule for the corresponding non serial schedule S.
- Serial Schedule: T2 → T3 → T1 → T4

Example -3

- Non Serial Schedule S.

T1	T2
R(A)	
A=A-10	
	R(A)
	Temp=0.2*A
	W(A)
	R(B)
W(A)	
R(B)	
B=B-10	
W(B)	
	B=B+Temp
	W(B)

- Find if the given schedule S is conflict serializable or not. Also find the serial schedules if any.

- **Trick:** Remove all computation steps. Because computation is done in memory(RAM) and no change has been done to the database stored in secondary memory.

Example -3

- Non Serial Schedule S.

Step	T1	T2
1	R(A)	
2		R(A)
3		W(A)
4		R(B)
5	W(A)	
6	R(B)	
7	W(B)	
8		W(B)

- Now, lets find conflicting pairs of operations.

1. $R_1(A) \rightarrow W_2(A)$ → T1 →T2
2. $R_2(A) \rightarrow W_1(A)$ → T2 →T1
3. $W_2(A) \rightarrow W_1(A)$ → T2 →T1
4. $R_2(B) \rightarrow W_1(B)$ → T2 →T1
5. $R_1(B) \rightarrow W_2(B)$ → T1 →T2

Example -3

- Non Serial Schedule S.

Step	T1	T2
1	R(A)	
2		R(A)
3		W(A)
4		R(B)
5	W(A)	
6	R(B)	
7	W(B)	
8		W(B)

- Lets add the first edge.

1. $R_1(A) \rightarrow W_2(A)$ → **T1 →T2**
2. $R_2(A) \rightarrow W_1(A)$ → T2 →T1
3. $W_2(A) \rightarrow W_1(A)$ → T2 →T1
4. $R_2(B) \rightarrow W_1(B)$ → T2 →T1
5. $R_1(B) \rightarrow W_2(B)$ → T1 →T2

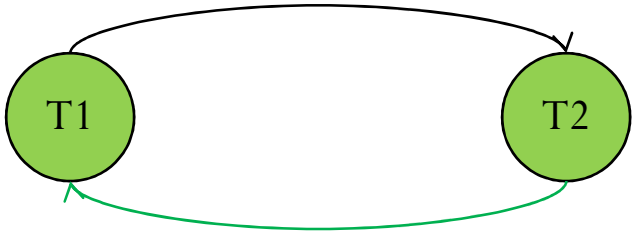
Example -3

- Non Serial Schedule S.

Step	T1	T2
1	R(A)	
2		R(A)
3		W(A)
4		R(B)
5	W(A)	
6	R(B)	
7	W(B)	
8		W(B)

- Now, lets add the second edge.

1. $R_1(A) \rightarrow W_2(A)$ → T1 →T2
2. $R_2(A) \rightarrow W_1(A)$ → **T2 →T1**
3. $W_2(A) \rightarrow W_1(A)$ → T2 →T1
4. $R_2(B) \rightarrow W_1(B)$ → T2 →T1
5. $R_1(B) \rightarrow W_2(B)$ → T1 →T2



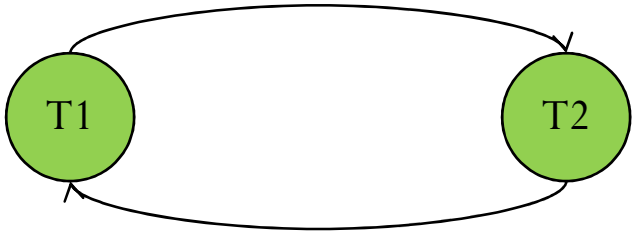
Example -3

- Non Serial Schedule S.

Step	T1	T2
1	R(A)	
2		R(A)
3		W(A)
4		R(B)
5	W(A)	
6	R(B)	
7	W(B)	
8		W(B)

- Rest of the edges are repetitive edges so, no need to add them again.

1. $R_1(A) \rightarrow W_2(A)$ → T1 →T2
2. $R_2(A) \rightarrow W_1(A)$ → T2 →T1
3. $W_2(A) \rightarrow W_1(A)$ → **T2 →T1**
4. $R_2(B) \rightarrow W_1(B)$ → **T2 →T1**
5. $R_1(B) \rightarrow W_2(B)$ → **T1 →T2**

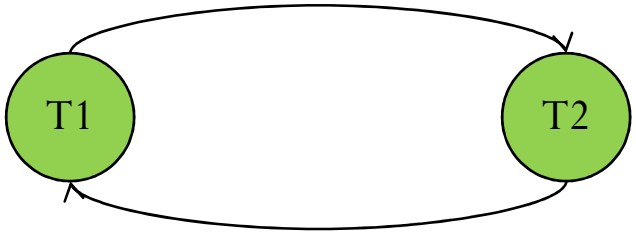


Example -3

- Non Serial Schedule S.

Step	T1	T2
1	R(A)	
2		R(A)
3		W(A)
4		R(B)
5	W(A)	
6	R(B)	
7	W(B)	
8		W(B)

- Clearly there is a cycle in the precedence graph.
- Thus, the given schedule is not conflict serializable.
- Thus, serial schedules are not possible.



Example -4

- Consider the following non serial schedule S.
- **S:** $R_1(A)$, $R_2(A)$, $R_1(B)$, $R_2(B)$, $R_3(B)$, $W_1(A)$, $W_2(B)$
- Find if the given schedule is conflict serializable or not. Also find the serial schedules if any.
- **Trick:** if you feel difficult to solve directly using the given format, then convert this format into table format.

Example -4

- Consider the following non serial schedule S.
- **S:** $R_1(A)$, $R_2(A)$, $R_1(B)$, $R_2(B)$, $R_3(B)$, $W_1(A)$, $W_2(B)$

Step	T1	T2	T3
1	R(A)		
2		R(A)	
3	R(B)		
4		R(B)	
5			R(B)
6	W(A)		
7		W(B)	

- Lets find the conflicting pairs of operations.

1. $R_2(A) \rightarrow W_1(A)$ → T2 →T1
2. $R_1(B) \rightarrow W_2(B)$ → T1 →T2
3. $R_3(B) \rightarrow W_2(B)$ → T3 →T2

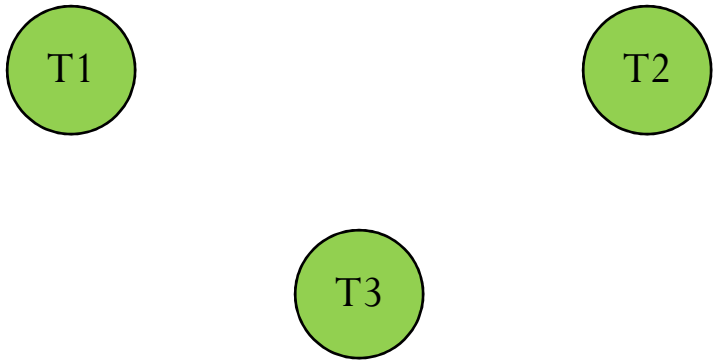
Example -4

- Consider the following non serial schedule S.
- **S:** $R_1(A)$, $R_2(A)$, $R_1(B)$, $R_2(B)$, $R_3(B)$, $W_1(A)$, $W_2(B)$

Step	T1	T2	T3
1	R(A)		
2		R(A)	
3	R(B)		
4		R(B)	
5			R(B)
6	W(A)		
7		W(B)	

- Lets create 3 nodes.

1. $R_2(A) \rightarrow W_1(A)$ → T2 →T1
2. $R_1(B) \rightarrow W_2(B)$ → T1 →T2
3. $R_3(B) \rightarrow W_2(B)$ → T3 →T2



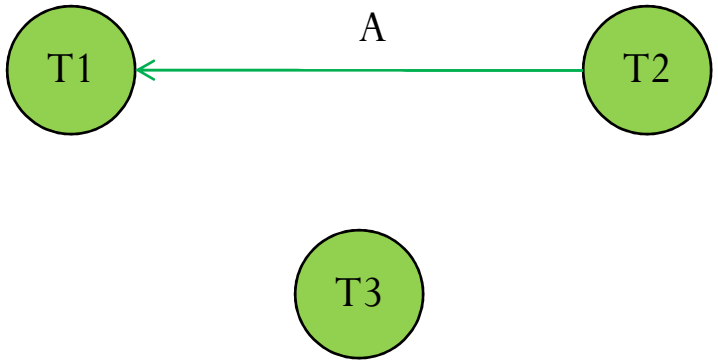
Example -4

- Consider the following non serial schedule S.
- **S:** $R_1(A), R_2(A), R_1(B), R_2(B), R_3(B), W_1(A), W_2(B)$

Step	T1	T2	T3
1	R(A)		
2		R(A)	
3	R(B)		
4		R(B)	
5			R(B)
6	W(A)		
7		W(B)	

- Lets add the first edge.

1. $R_2(A) \rightarrow W_1(A)$ → **T2 →T1**
2. $R_1(B) \rightarrow W_2(B)$ → T1 →T2
3. $R_3(B) \rightarrow W_2(B)$ → T3 →T2



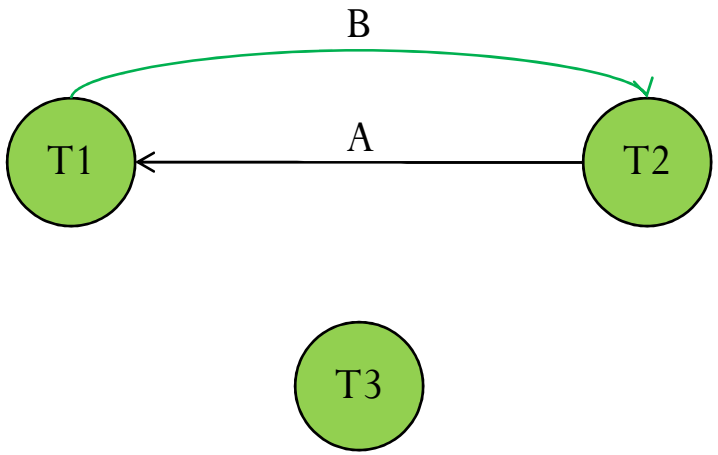
Example -4

- Consider the following non serial schedule S.
- **S:** $R_1(A), R_2(A), R_1(B), R_2(B), R_3(B), W_1(A), W_2(B)$

Step	T1	T2	T3
1	R(A)		
2		R(A)	
3	R(B)		
4		R(B)	
5			R(B)
6	W(A)		
7		W(B)	

- Lets add the second edge.

1. $R_2(A) \rightarrow W_1(A)$ → T2 →T1
2. $R_1(B) \rightarrow W_2(B)$ → **T1 →T2**
3. $R_3(B) \rightarrow W_2(B)$ → T3 →T2



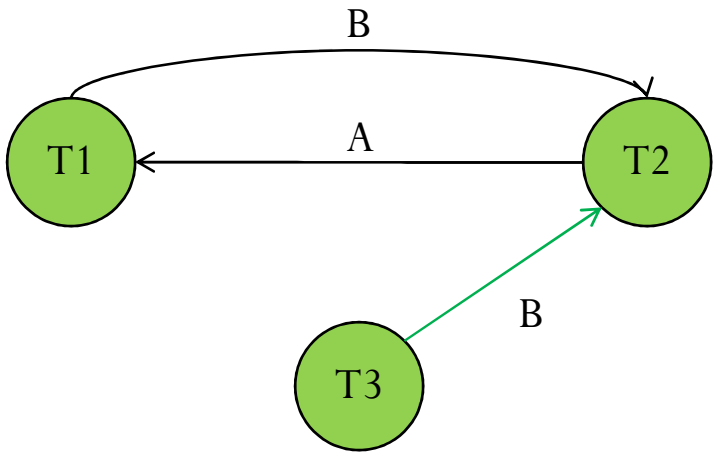
Example -4

- Consider the following non serial schedule S.
- **S:** $R_1(A), R_2(A), R_1(B), R_2(B), R_3(B), W_1(A), W_2(B)$

Step	T1	T2	T3
1	R(A)		
2		R(A)	
3	R(B)		
4		R(B)	
5			R(B)
6	W(A)		
7		W(B)	

- Lets add the third edge.

1. $R_2(A) \rightarrow W_1(A)$ → T2 →T1
2. $R_1(B) \rightarrow W_2(B)$ → T1 →T2
3. $R_3(B) \rightarrow W_2(B)$ → **T3 →T2**



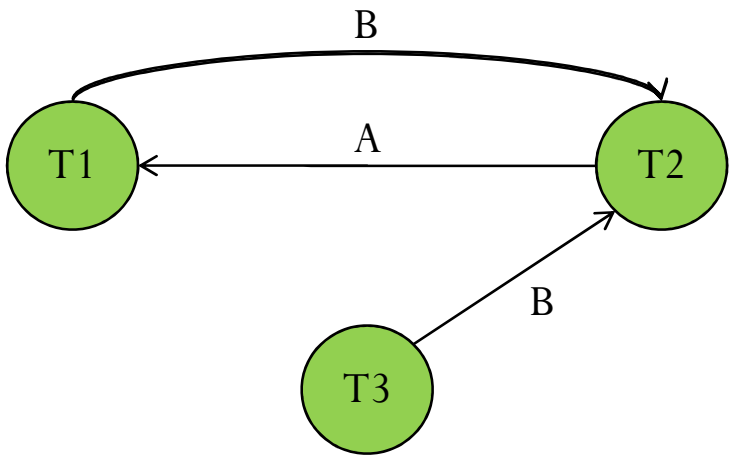
Example -4

- Consider the following non serial schedule S.
- **S:** $R_1(A)$, $R_2(A)$, $R_1(B)$, $R_2(B)$, $R_3(B)$, $W_1(A)$, $W_2(B)$

Step	T1	T2	T3
1	R(A)		
2		R(A)	
3	R(B)		
4		R(B)	
5			R(B)
6	W(A)		
7		W(B)	

- Clearly, the final precedence graph is having a cycle. So, the given schedule is not conflict serializable. Thus, no serial schedule is possible.

[**NOTE:** It does not matter if the cycle formed contains different data items(A,B). The cycle between T1 & T2 means that one edge(A) is saying that T2 should execute completely before T1 is executed. And the other edge(B) is saying that T1 should execute completely before T2 is executed. Both of these conditions are not possible at the same time.]



Wait For Graph

- It is one of the methods for detecting the deadlock situation in a database.
- It is suitable for small databases. For big databases it is better to prevent deadlock from happening rather than rectifying a deadlock situation.
- In this method, a graph is drawn based on the transaction and their lock on resources.
- If the graph created has a closed loop or a cycle, then there is a deadlock.
- There are mainly 2 types of locks: Shared lock S(A) & Exclusive lock X(A).
- Conflicting locks: S-X , X-S & X-X
- Non conflicting lock: S-S

Example -1

- Consider the following schedule.

T1	T2	T3	T4
S(A)			
S(D)			
	X(B)		
S(B)			
		S(D)	
		S(C)	
	X(C)		
			X(B)
		X(A)	

- Find whether a deadlock exists in the system or not.

Example -1

- Consider the following schedule.

Step	T1	T2	T3	T4
1	S(A)			
2	S(D)			
3		X(B)		
4	S(B)			
5			S(D)	
6			S(C)	
7		X(C)		
8				X(B)
9			X(A)	

- For each step will have to find whether the given lock is conflicting with any of the previous locks made by other transactions.

1. In step 1, shared lock S(A) will not have any conflict as no other locks exists in the system as of now.
2. In step 2, shared lock S(D) will not conflict with the any other lock as other transactions have not acquired any lock as of now.
3. In Step 3, exclusive lock X(B) by T2 will not conflict with previous locks by T1 as they are on different data items.
4. In Step 4, shared lock S(B) by T1 may conflict with a previous exclusive lock X(B) held by T2. So, T1 will have to wait for T2 to release its lock on data item B. Thus, T1 → T2

Example -1

- Consider the following schedule.

Step	T1	T2	T3	T4
1	S(A)			
2	S(D)			
3		X(B)		
4	S(B)			
5			S(D)	
6			S(C)	
7		X(C)		
8				X(B)
9			X(A)	

5. In step 5, shared lock S(D) by T3 will not conflict with a previous shared lock S(D) held by T1 (as shared locks do not conflict).
6. In step 6, shared lock S(C) by T3 will not conflict with any other previous locks by other transactions.
7. In step 7, exclusive lock X(C) by T2 may conflict with a previous shared lock S(C) held by T3.. So, T2 will have to wait for T3 to release its lock, so that T2 can acquire its lock. Thus, T2→T3
8. In step 8, exclusive lock X(B) by T4 will conflict with the shared lock S(B) held T1. Thus, T4→T1. Note that X(B) by T4 will not conflict with X(B) by T2, because the by time we reach step 8 T2 had released its lock and the last lock on data item B is now with transaction T1.

Example -1

- Consider the following schedule.

Step	T1	T2	T3	T4
1	S(A)			
2	S(D)			
3		X(B)		
4	S(B)			
5			S(D)	
6			S(C)	
7		X(C)		
8				X(B)
9			X(A)	

9. In step 9, .exclusive lock X(A) by T3 may conflict with the last lock on A which is a shared lock S(A) held by T1. Thus T3 will have to wait for T1 to release its lock on data item A. Thus T3→T1.

- So, we have following conflicts or waits:

1. T1→T2
2. T2→T3
3. T4→T1
4. T3→T1

Now we can plot the respective edges on the graph.

Example -1

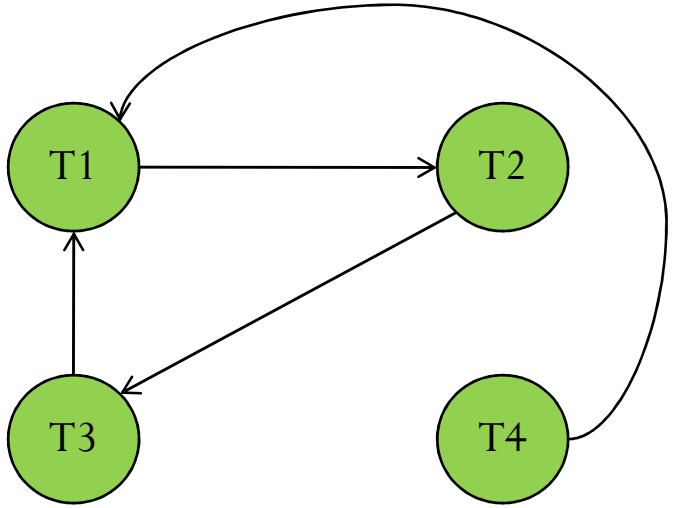
- Consider the following schedule.

Step	T1	T2	T3	T4
1	S(A)			
2	S(D)			
3		X(B)		
4	S(B)			
5			S(D)	
6			S(C)	
7		X(C)		
8				X(B)
9			X(A)	

- So, we have following conflicts or waits:

1. T1→T2
2. T2→T3
3. T4→T1
4. T3→T1

Now we can plot the respective edges on the graph.

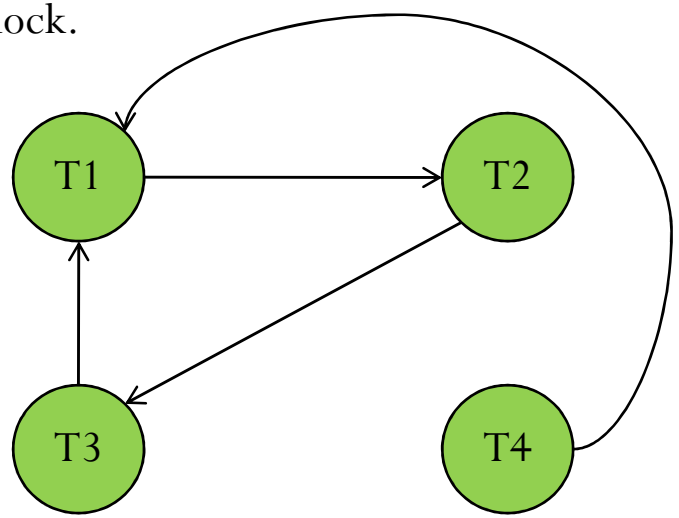


Example -1

- Consider the following schedule.

Step	T1	T2	T3	T4
1	S(A)			
2	S(D)			
3		X(B)		
4	S(B)			
5			S(D)	
6			S(C)	
7		X(C)		
8				X(B)
9			X(A)	

- Clearly there is a cycle in T1, T2 and T3 transactions. Thus, this system may have a deadlock.
- If we want to get this system out of deadlock, then we will have to break the cycle. To do this we will have to abort one of the transactions among T1, T2 & T3. Thus resources held by that transaction will be release and system will get out of deadlock.



Exxample-2

- Given following schedule:

T1	T2	T3
S(B)		
R(B)		
		S(A)
		R(A)
X(A)		
	S(C)	
	R(C)	
	X(B)	
		X(C)

- Find whether the given schedule may have a deadlock or not.

Exxample-2

- Given following schedule:

Step	T1	T2	T3
1	S(B)		
2	R(B)		
3			S(A)
4			R(A)
5	X(A)		
6		S(C)	
7		R(C)	
8		X(B)	
9			X(C)

- Lets find the conflicting pairs of lock, so that we can draw wait for graph.

1. In step 1, no wait is required as all resources are free, so this lock will be granted immediately.
2. Step 2 is a read operation, no need to check it.
3. In step 3, shared lock S(A) is required by T3 which will be granted immediately as the resource is free.
4. Step 4 is read operation, so ignore it.
5. In step 5, exclusive lock X(A) is requested by T1 which may conflict with a previous shared lock S(A) held by T3. So, T1 will have to wait for T3 to release its lock. Thus, T1→T3.
6. In step 6, a shared lock is S(C) is requested by T2, which will be granted immediately as the resource is free.

Exxample-2

- Given following schedule:

Step	T1	T2	T3
1	S(B)		
2	R(B)		
3			S(A)
4			R(A)
5	X(A)		
6		S(C)	
7		R(C)	
8		X(B)	
9			X(C)

7. Step 7 is a read operation, so ignore it.
8. In step 8, an exclusive lock X(B) is requested by T2, which may conflict with the previous shared lock S(B) held by T1. So, T2 will have to wait for T1 to release its lock on data item B. Thus, T2→T1
9. In step 9, an exclusive lock X(C) is requested by T3 which may conflict with the previous shared lock S(C) held by T2. So, T3 will have to wait for T2 to release its lock on data item C. Thus, T3→T2.

Exxample-2

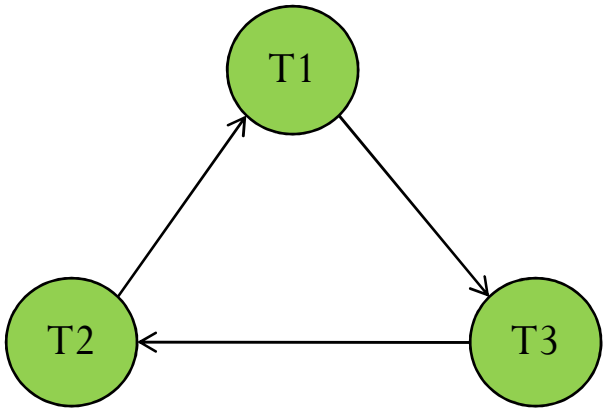
- Given following schedule:

Step	T1	T2	T3
1	S(B)		
2	R(B)		
3			S(A)
4			R(A)
5	X(A)		
6		S(C)	
7		R(C)	
8		X(B)	
9			X(C)

- So we have the following wait for pairs:

1. T1→T3
2. T2→T1
3. T3→T2

- Lets draw the graph for this now:



Exxample-2

- Given following schedule:

Step	T1	T2	T3
1	S(B)		
2	R(B)		
3			S(A)
4			R(A)
5	X(A)		
6		S(C)	
7		R(C)	
8		X(B)	
9			X(C)

- Clearly, there is a cycle in the wait for graph. Thus, a deadlock is possible.