Transaction Management

Concurrency Control (1)

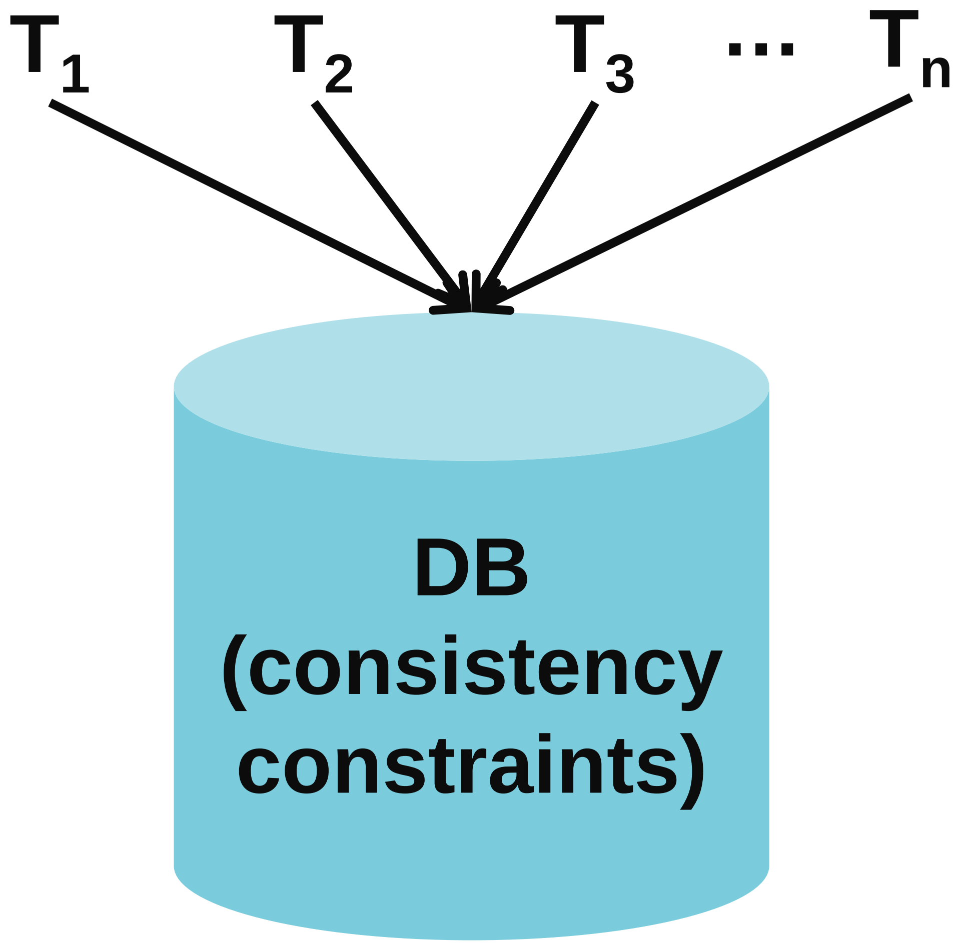
Review: ACID Properties

- Atomicity
  - Actions are never left partially executed
- Consistency
  - Actions leave the DB in a consistent state
- Isolation
  - Actions are not affected by other concurrent actions
- Durability
  - Effects of completed actions are resilient against system failures

Chapter 18 Concurrency Control

Desired Effect: Transactions not affected by other concurrent transactions.

A
C
Isolation
D



Example

T_1:	Read(A)	T_2:	Read(A)
	$A \leftarrow A+100$		$A \leftarrow A \times 2$
	Write(A)		Write(A)
	Read(B)		Read(B)
	$B \leftarrow B+100$		$B \leftarrow B \times 2$
	Write(B)		Write(B)

Constraint: A=B

Schedule A

T_1	T_2	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
Read(B); B ← B+100;			
Write(B);			125
	Read(A); A ← A×2;		
	Write(A);	250	
	Read(B); B ← B×2;		
	Write(B);		250
		250	250

No interleaving of actions of transactions. (Serial Schedule)

Schedule B

T_1	T_2	A	B
		25	25
	Read(A); A ← A×2;		
	Write(A);	50	
	Read(B); B ← B×2;		
	Write(B);		50
Read(A); A ← A+100			
Write(A);		150	
Read(B); B ← B+100;			
Write(B);			150
		150	150

No interleaving of actions of transactions. (Serial Schedule)

Schedule C

T_1	T_2	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
	Read(A); A ← A×2;		
	Write(A);	250	
Read(B); B ← B+100;			
Write(B);			125
	Read(B); B ← B×2;		
	Write(B);		250
		250	250

Serializable, but not serial schedule

Schedule D

T_1	T_2	A 25	B 25
Read(A); A ← A+100			
Write(A);		125	
	Read(A); A ← A×2;		
	Write(A);	250	
	Read(B); B ← B×2;		
	Write(B);		50
Read(B); B ← B+100;			
Write(B);			150
		250	150

Nonserializable schedule

Schedule E

Same as Schedule D, but with new T_2'

T_1	T_2'	A	B
		25	25
Read(A); A ← A+100			
Write(A);		125	
	Read(A); A ← A×1;		
	Write(A);	125	
	Read(B); B ← B×1;		
	Write(B);		25
Read(B); B ← B+100;			
Write(B);			125
		125	125

Serializable schedule (due to details of actions)

Comments

- What schedules that are "good", regardless of
  - Initial state;
  - Transaction semantics
- Only look at order of read and writes

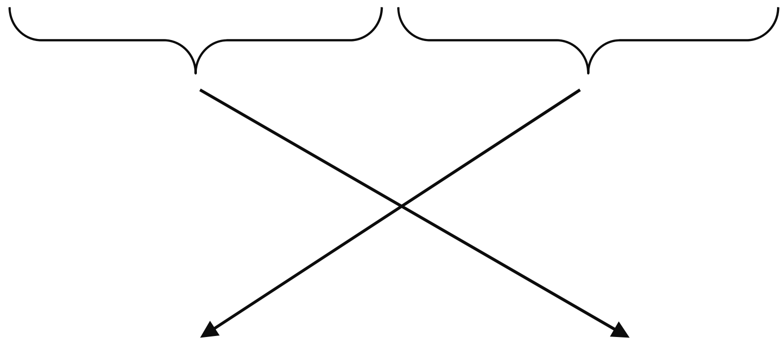
Example:

$$S(C) = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$$

Comments (cont.)

Example:

$S(C) = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$



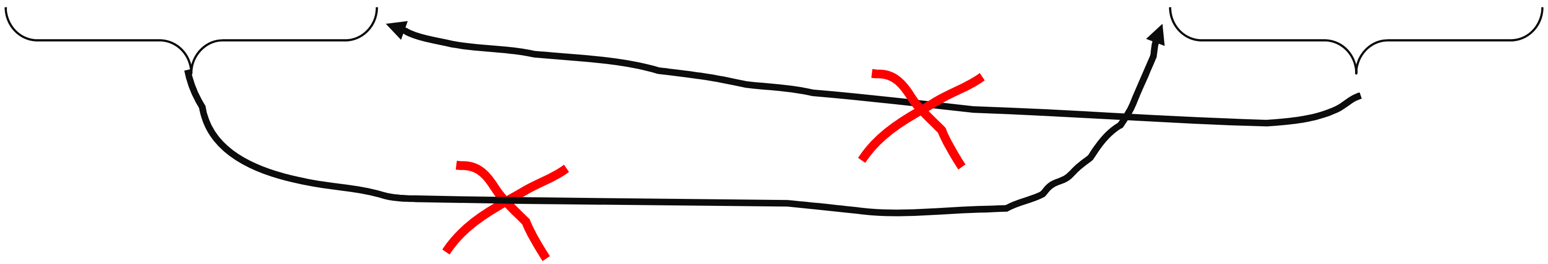
S(C)’ = r1(A)w1(A) r1(B)w1(B)r2(A)w2(A)r2(B)w2(B)

T_1 T_2

Comments (cont.)

- However, for Schedule D:

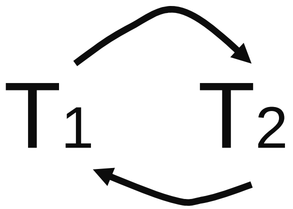
S(D) = r1(A)w1(A)r2(A)w2(A)r2(B)w2(B)r1(B)w1(B)



Thus, T_2 must precede T_1 in any equivalent schedule, i.e., $T_2 \rightarrow T_1$

Comments (cont.)

- $T_2 \rightarrow T_1$
- Also, $T_1 \rightarrow T_2$



⇒ S(D) cannot be rearranged into a serial schedule

⇒ S(D) is not "equivalent" to any serial schedule

⇒ S(D) is "bad"

Comments (cont.)

Returning to S(C)

$$S(C) = r_1(A)w_1(A)r_2(A)w_2(A)r_1(B)w_1(B)r_2(B)w_2(B)$$

$T_1 \rightarrow T_2$ $\quad$ $T_1 \rightarrow T_2$

No Cycles! ⇒ **S(C) is "equivalent" to a serial schedule (in this case, T_1,T_2)**

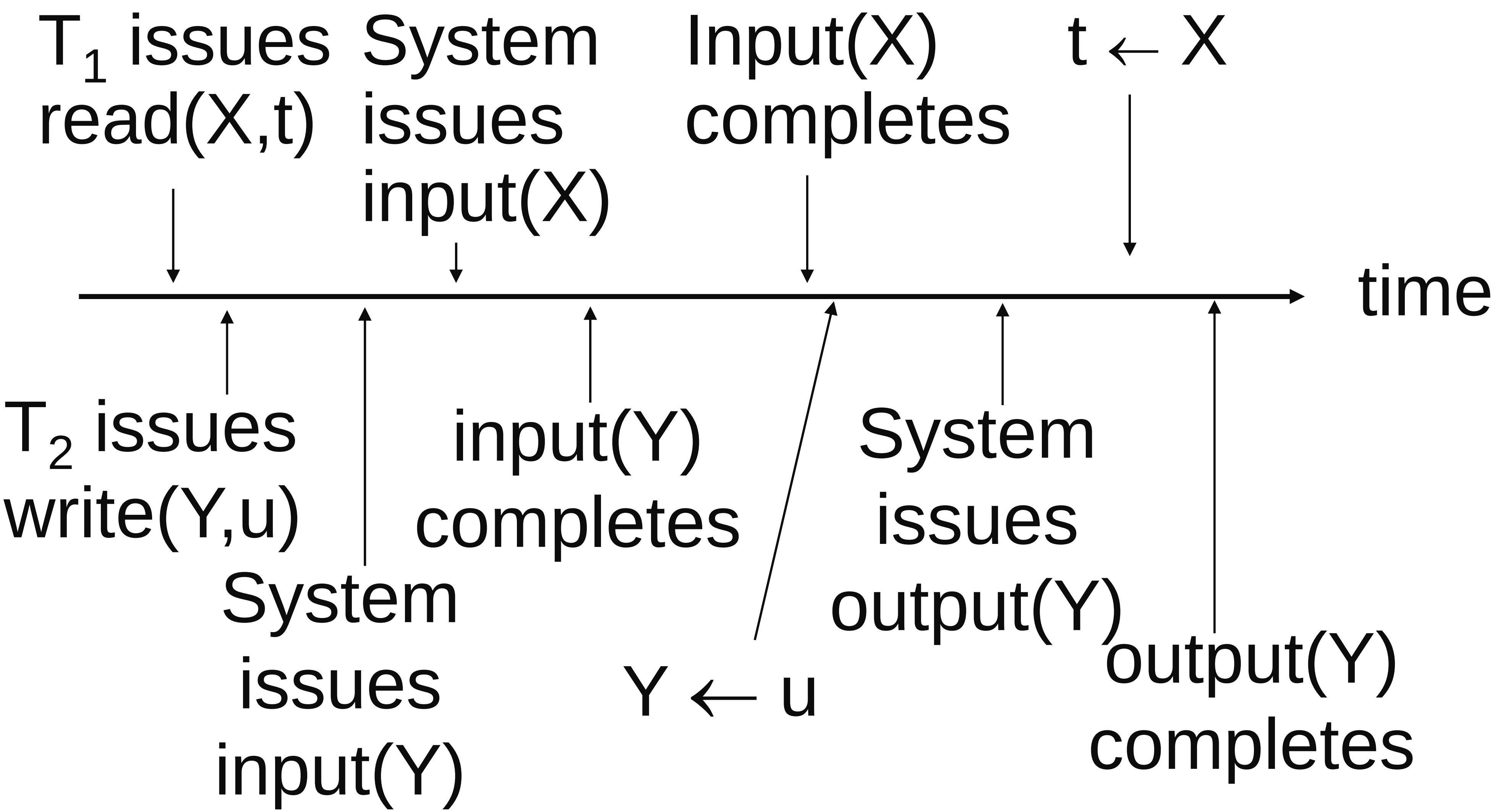
Concepts

- Transaction: sequence of $r_i(X)$, $w_i(X)$ actions
- Conflicting actions:

r1(A)	w2(A)	w1(A)
w2(A)	r1(A)	w2(A)

- Schedule: represents chronological order in which actions are executed.
- Serial schedule: no interleaving of actions or transactions.
- Serializable schedule: there exists a serial schedule such that for every initial database state, the two schedules are the same.

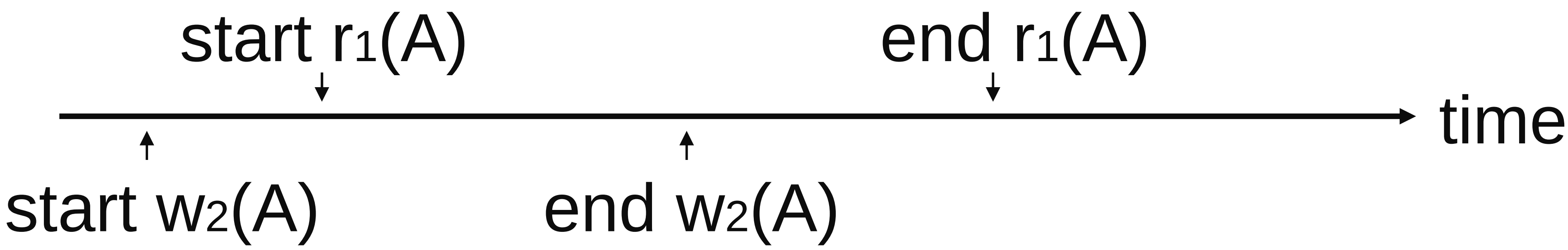
Concurrent Actions?



Concurrent Actions? (cont.)

- So net effect is either
  - S = …r1(X)…w2(Y)… or
  - S = …w2(Y)…r1(X)…

Conflicting, Concurrent Actions on Same Object?



- Assume equivalent to either r1(A)w2(A) or w2(A)r1(A)
- Low level synchronization mechanism
- Assumption called “atomic actions”

Conflict Equivalent/Serializable

- Definitions:

S_1, S_2 are conflict equivalent schedules if S_1 can be transformed into S_2 by a series of swaps on non-conflicting actions.

A schedule is conflict serializable if it is conflict equivalent to some serial schedule.

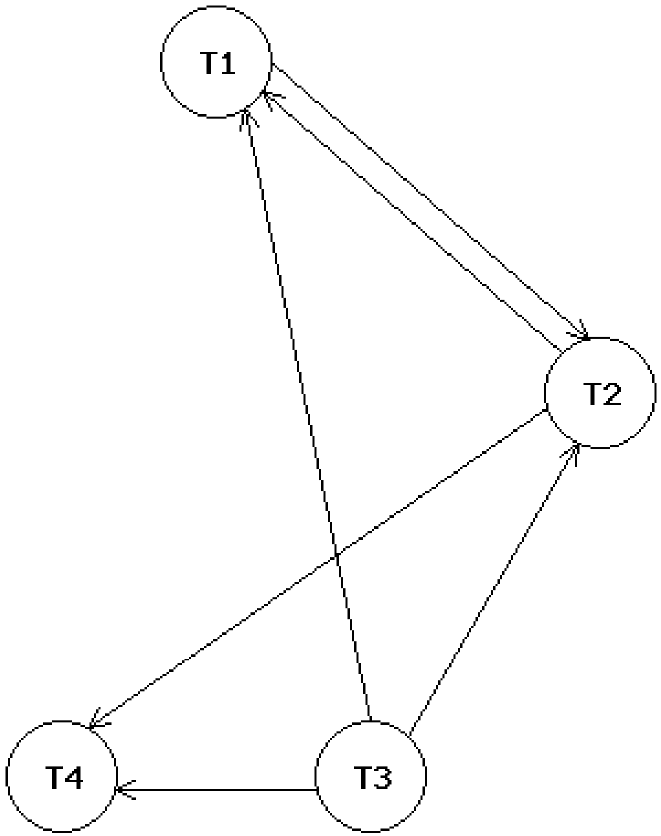
Precedence Graph P(S)

- S is a schedule
- In the graph:
  - Nodes: transactions in S;
  - Arcs: $T_i \rightarrow T_j$ whenever
    - $p_i(A)$, $q_j(A)$ are actions in S involving the same database element;
    - $p_i(A) <_S q_j(A)$ ($p_i(A)$ is ahead of $q_j(A)$ in S);
    - At least one of p_i, q_j is a “write” action.

Examples (1)

- What is P(S) for

S=w3(A)w2(C)r1(A)w1(B)r1(C)w2(A)r4(A)w4(D)

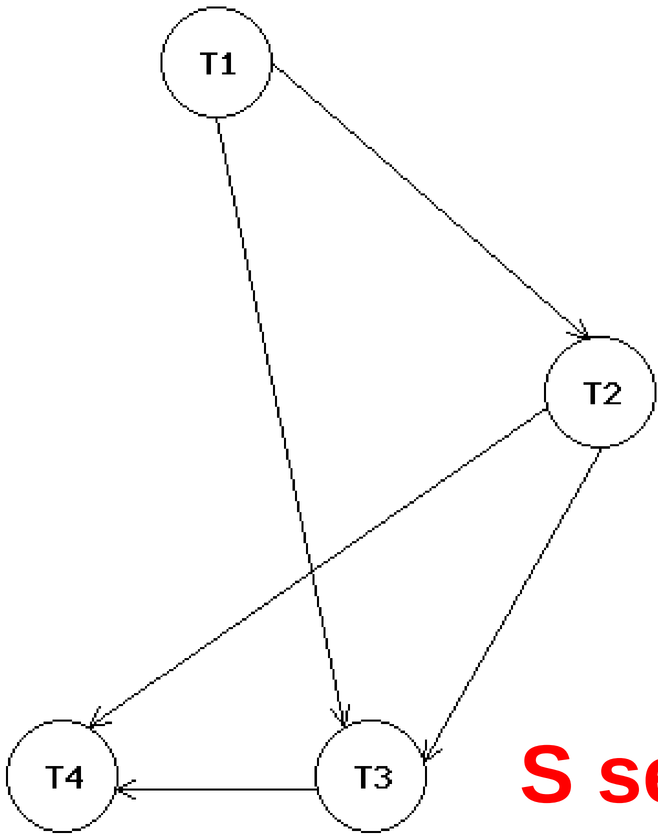


Is S serializable?

Examples (2)

- What is P(S) for

S=r1(A)w1(B)r1(C)w2(C)w2(A)w3(A)r4(A)w4(D)



S serial → P(S) acyclic!

To-Do-List

- Do a research on Precedence Graph. Can you write a program to draw the graph?
- What is the usage of Precedence Graph?