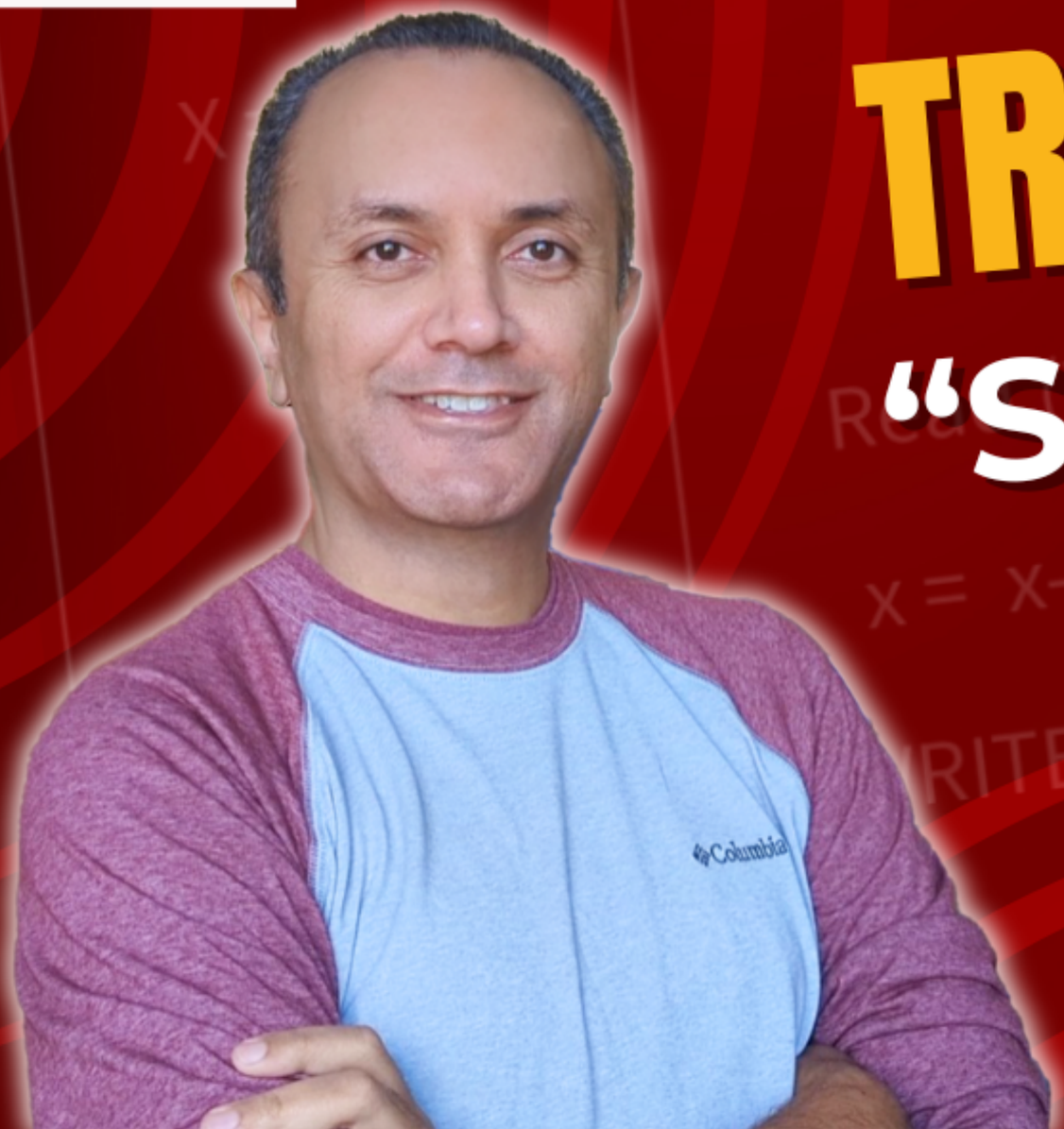
TECH
VAULT



TRANSACTIONS

"Schedules"

AMR ELHELW

ACID Properties

A — Atomicity

Each transaction is “all or nothing”

C — Consistency

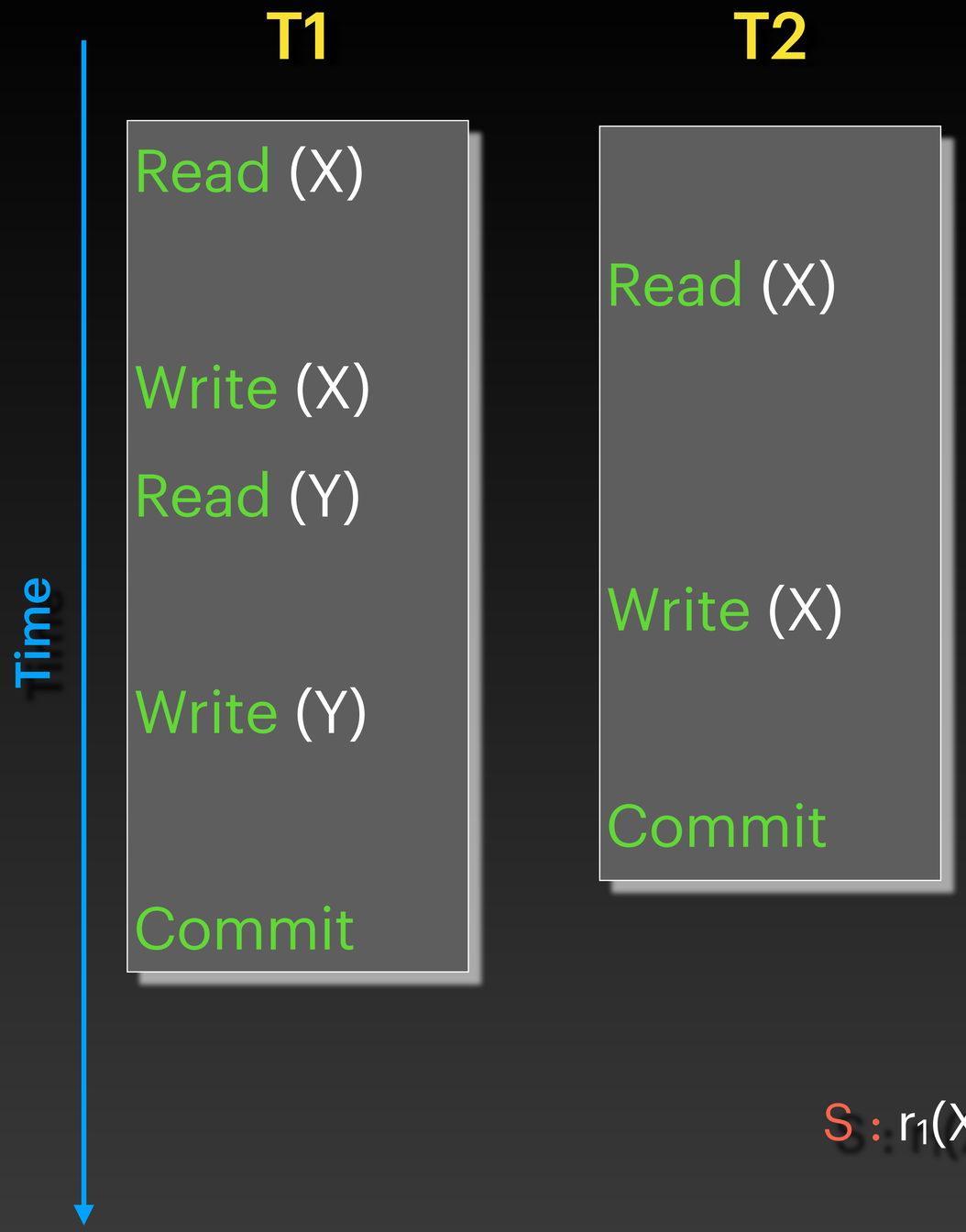
Data should be valid according to all defined rules

I — Isolation

Transactions do not affect each other

D — Durability

Committed data would not be lost, even after power failure.

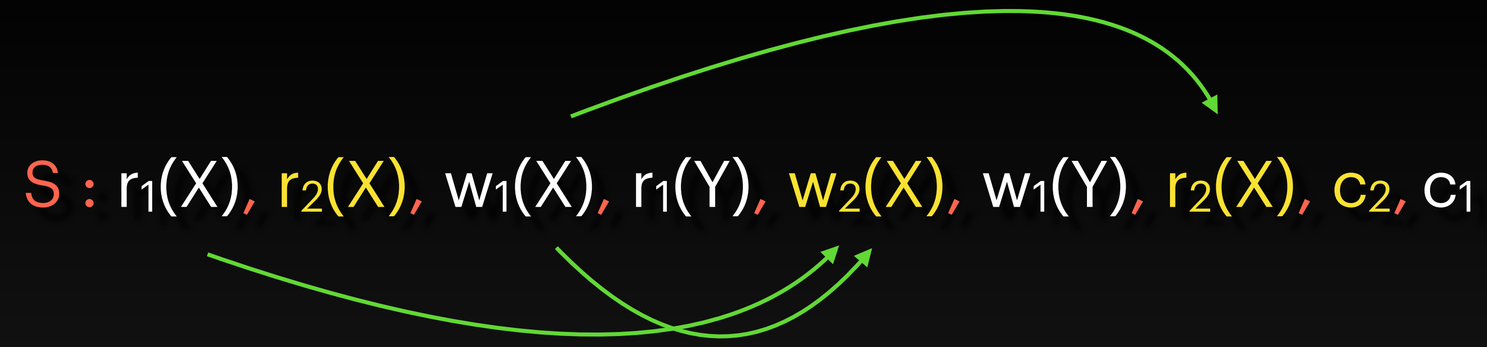


S : $r_1(X)$, $r_2(X)$, $w_1(X)$, $r_1(Y)$, $w_2(X)$, $w_1(Y)$, c_2, c_1

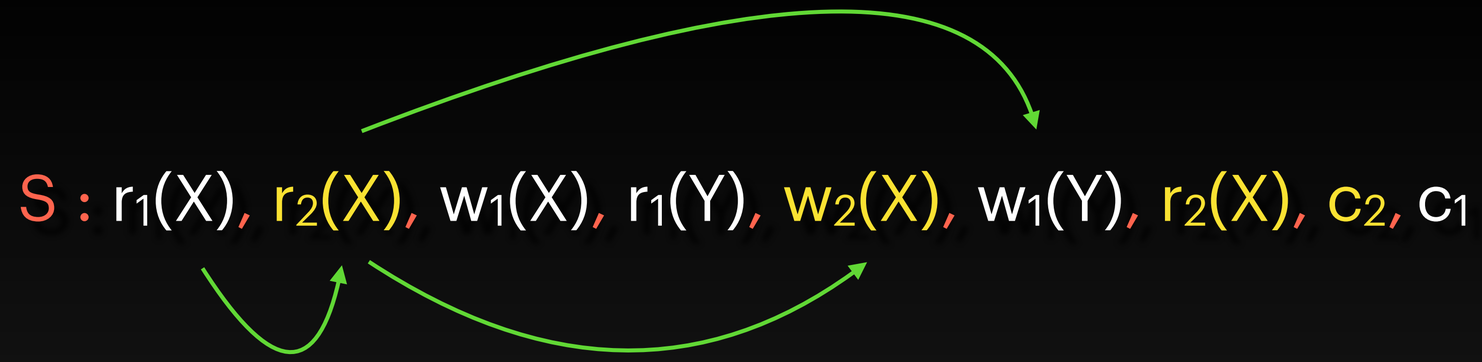
Conflicting Operations

Conditions for 2 conflicting operations:

- They belong to different transactions
- They access the same object
- At least one of them is a write operation



- Read-Write (R-W) conflict
- Write-Read (W-R) conflict
- Write-Write (W-W) conflict



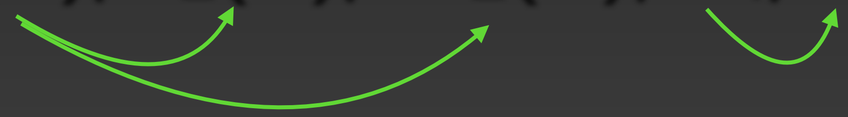
- Same object, diff transactions, but both reads
- Read and write, diff transaction, but diff objects
- Read and write, same object, but same transaction

Recoverable Schedule

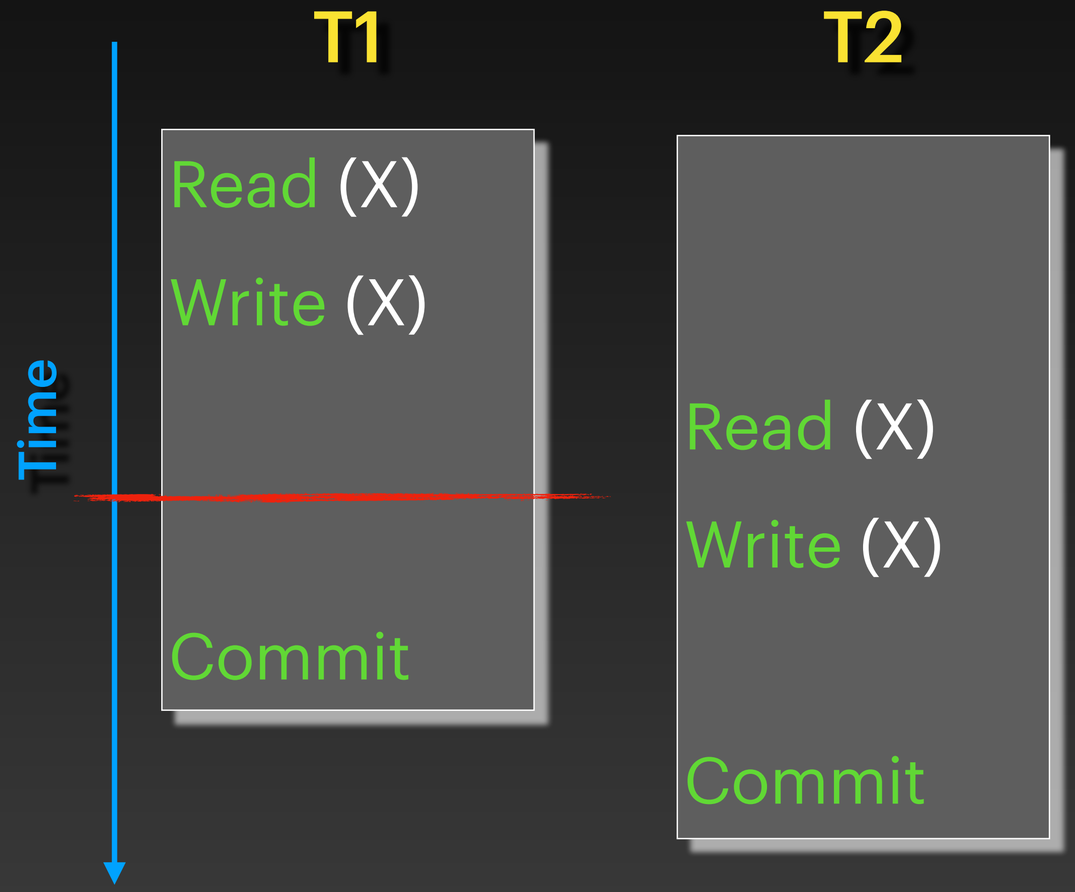
- If T1 writes X then T2 reads/writes X
- Then T1 must commit before T2 commits

Look for the W-R and W-W conflicts — the commits must be in the same order

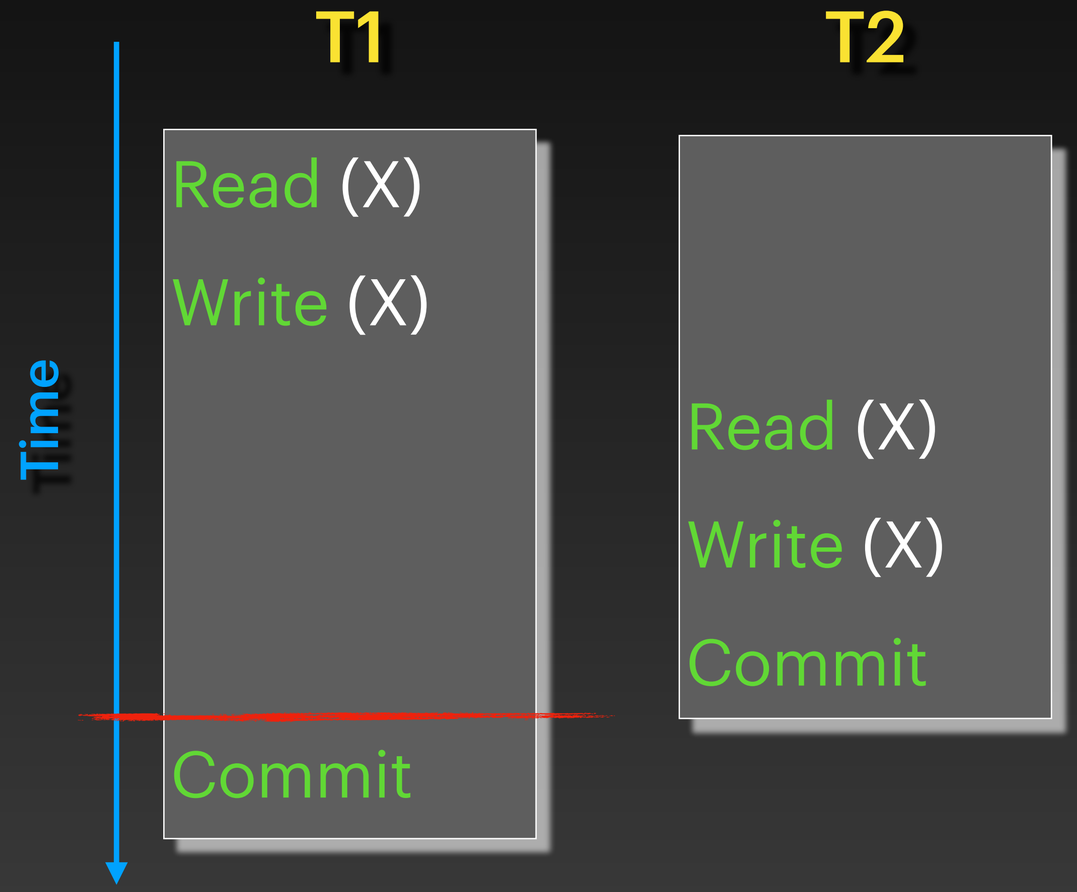
S : $r_1(X)$, $w_1(X)$, $r_2(X)$, $w_2(X)$, c_1, c_2



S : $r_1(X)$, $w_1(X)$, $r_2(X)$, $w_2(X)$, c_1, c_2



S : $r_1(X)$, $w_1(X)$, $r_2(X)$, $w_2(X)$, c_2, c_1



Cascadeless Schedule

- Subset of recoverable schedules, with no cascading rollback

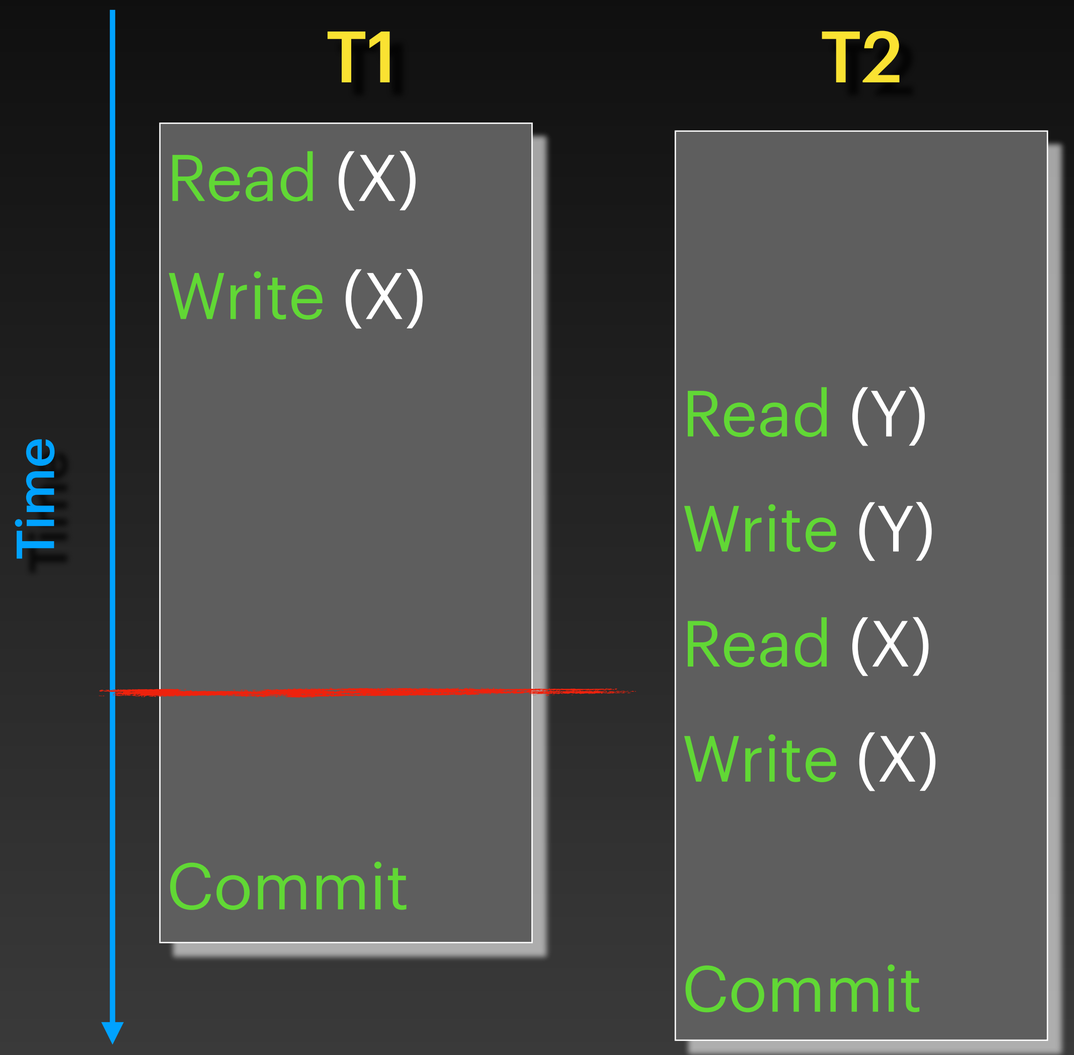
- *Look for the W-R and W-W conflicts — the commits must be in the same order*
- *For any W-R conflicts, the first transaction commits before the second reads*

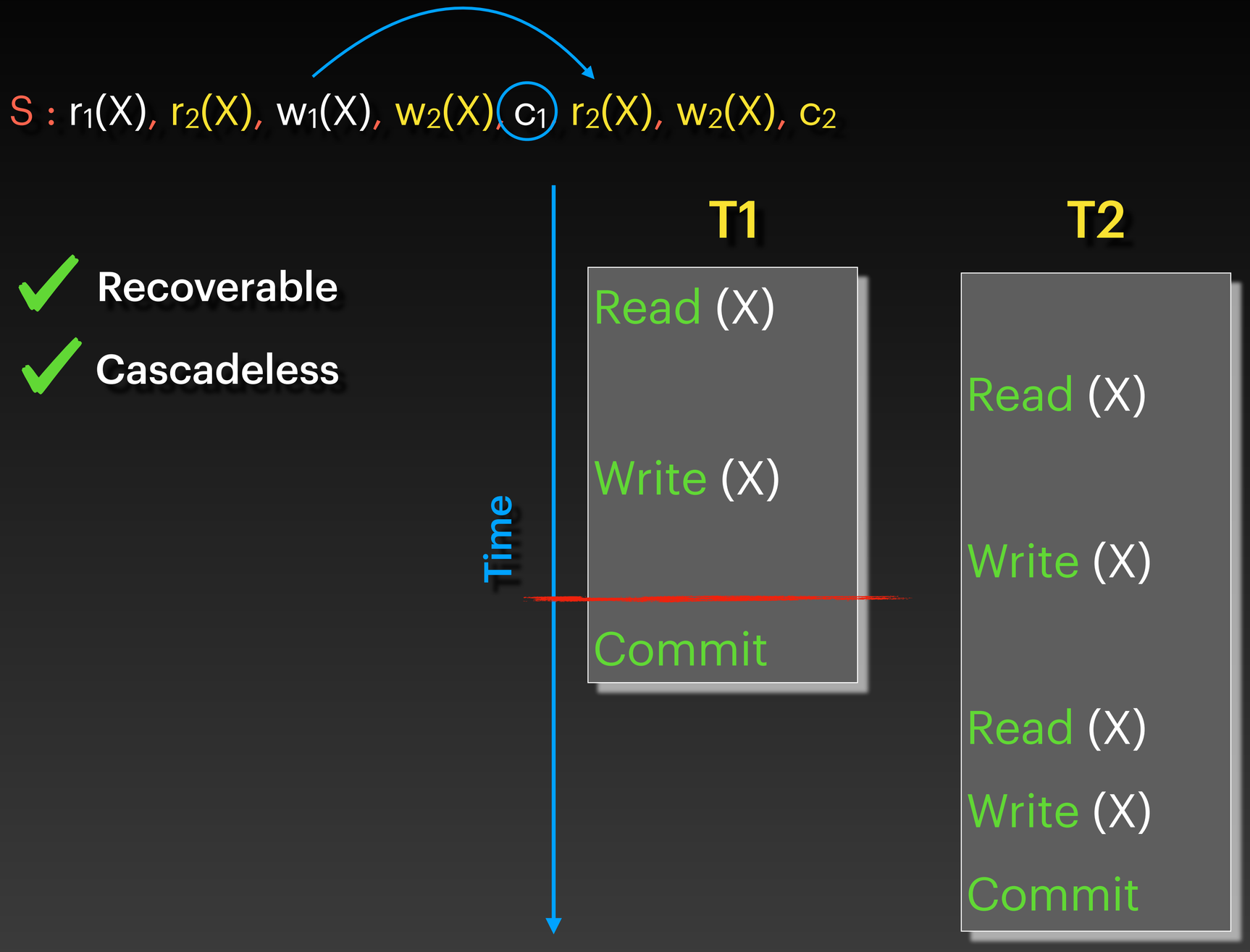
T1 not committed yet

S : $r_1(X)$, $w_1(X)$, $r_2(Y)$, $w_2(Y)$, $r_2(X)$, $w_2(X)$, c_1, c_2

✓ Recoverable

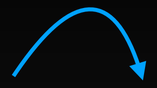
✗ Cascadeless





Strict Schedule

- Subset of cascadeless schedules, with stricter conditions
  - *Look for the W-R and W-W conflicts — the commits must be in the same order*
  - ~~*For any W-R conflicts, the first transaction commits before the second reads*~~
  - *For any W-R or W-W conflicts, the first transaction commits before the second reads/writes.*



S1 : $r_1(X)$, $r_2(X)$, $w_1(X)$, $w_2(X)$, c_1, $r_2(X)$, $w_2(X)$, c_2

✓ Recoverable

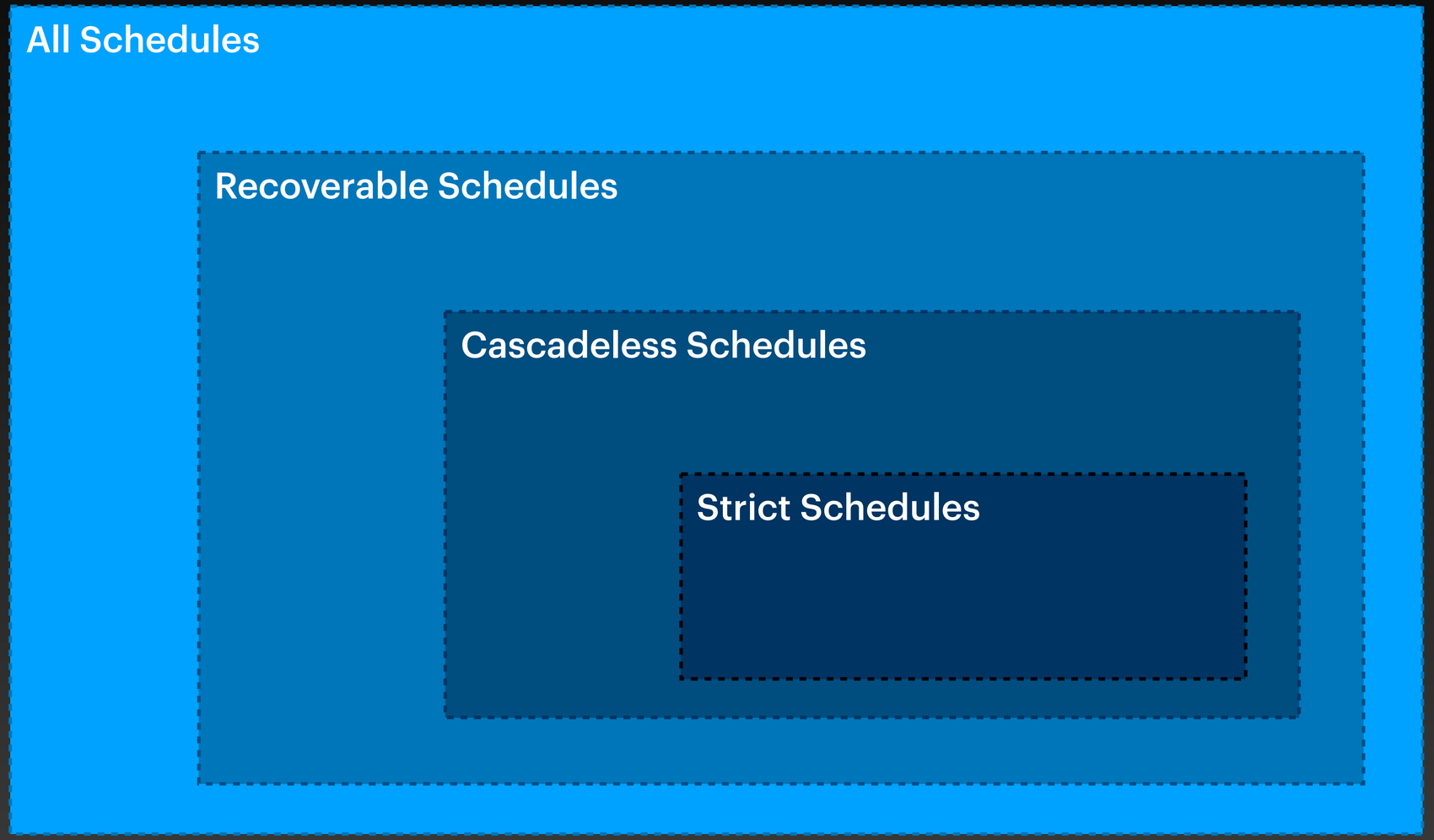
✓ Cascadeless

✗ Strict

S2 : $r_1(X)$, $r_2(X)$, $w_1(X)$, c_1, $w_2(X)$, $r_2(X)$, $w_2(X)$, c_2

✓ Strict

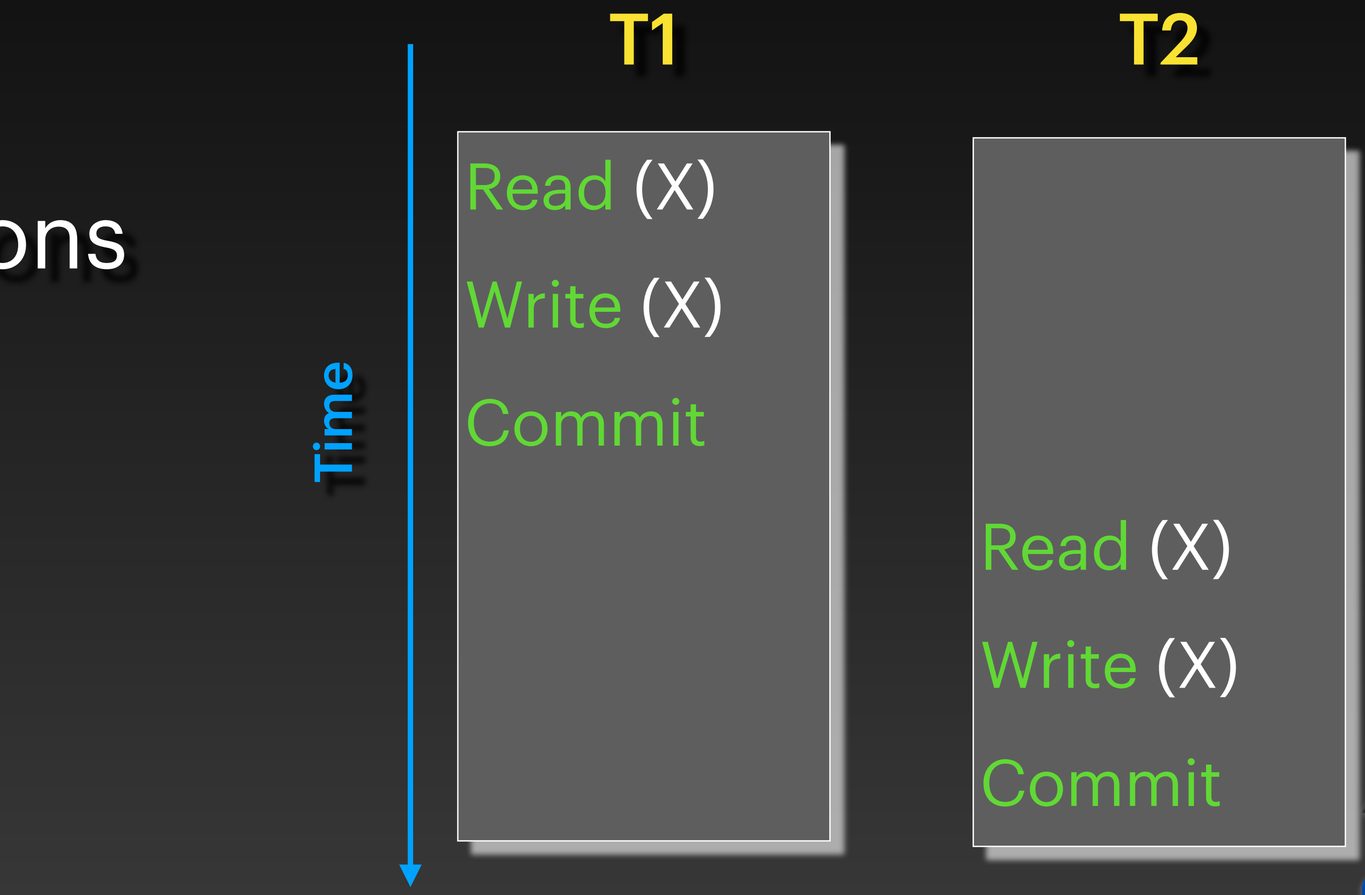
Schedule Recoverability



Serial Schedule

- No interleaving of operations from different transactions

- *n* transactions —> n! possible serial schedules



Serializable Schedule

- A schedule that is “equivalent” to a serial schedule
  - Several ways to define equivalence
  - One type of equivalence is “conflict equivalence”

Conflict Equivalence

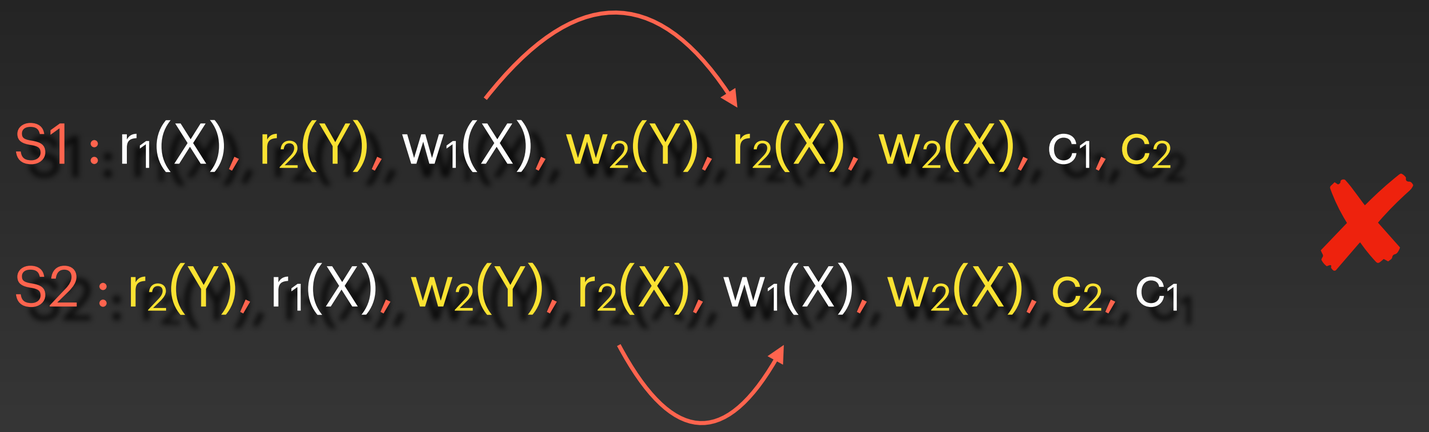
- Two schedules are conflict equivalent if the relative order of any two conflicting operations is the same in both schedules

S1 : $r_1(X)$, $r_2(Y)$, $w_1(X)$, $w_2(Y)$, $r_2(X)$, $w_2(X)$, c_1, c_2

S2 : $r_2(Y)$, $r_1(X)$, $w_2(Y)$, $w_1(X)$, $r_2(X)$, $w_2(X)$, c_2, c_1

Conflict Equivalence

- Two schedules are conflict equivalent if the relative order of any two conflicting operations is the same in both schedules



Serializable Schedule

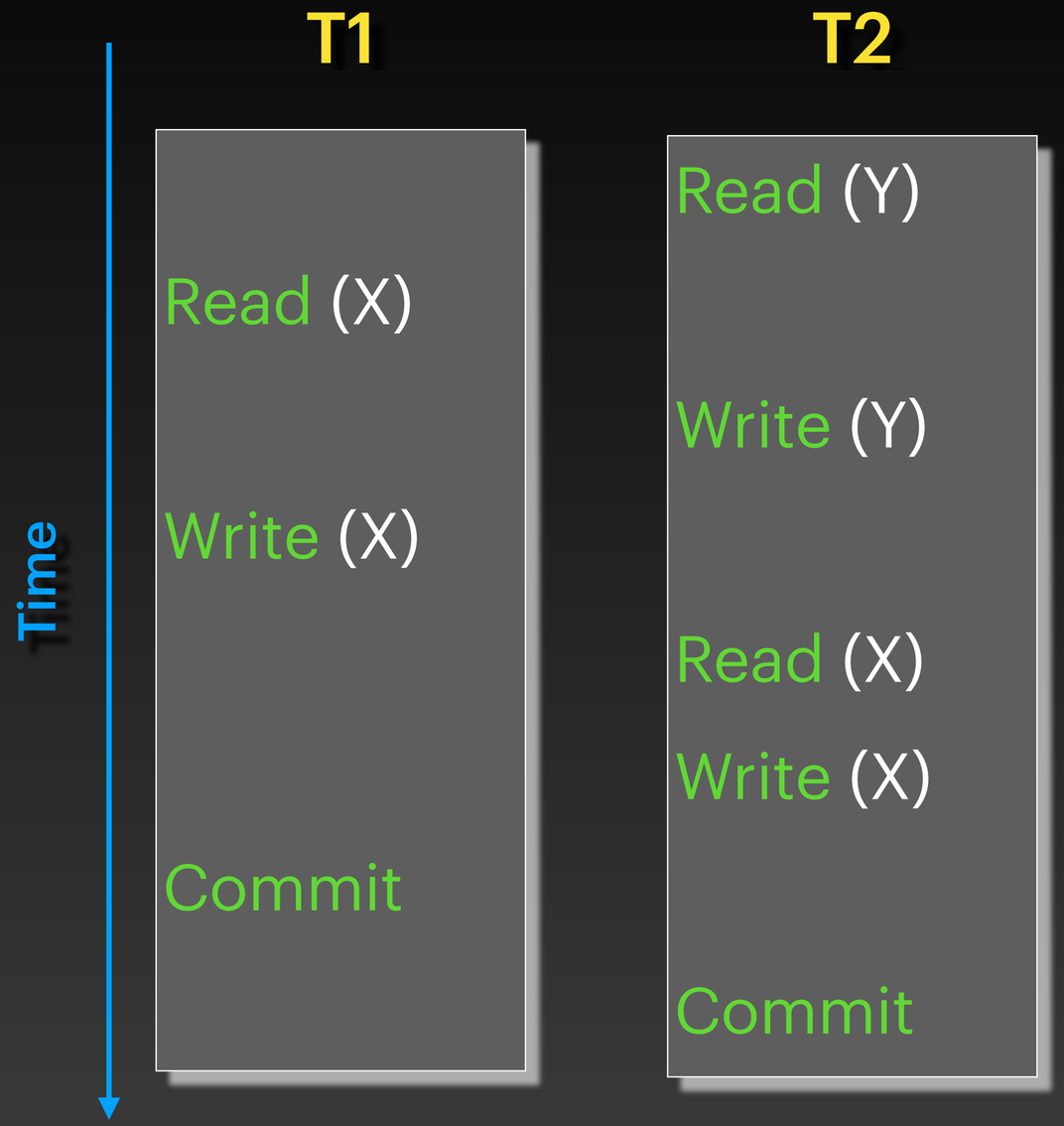
- A schedule that is conflict equivalent to a serial schedule is Conflict Serializable.

Conflict Equivalent

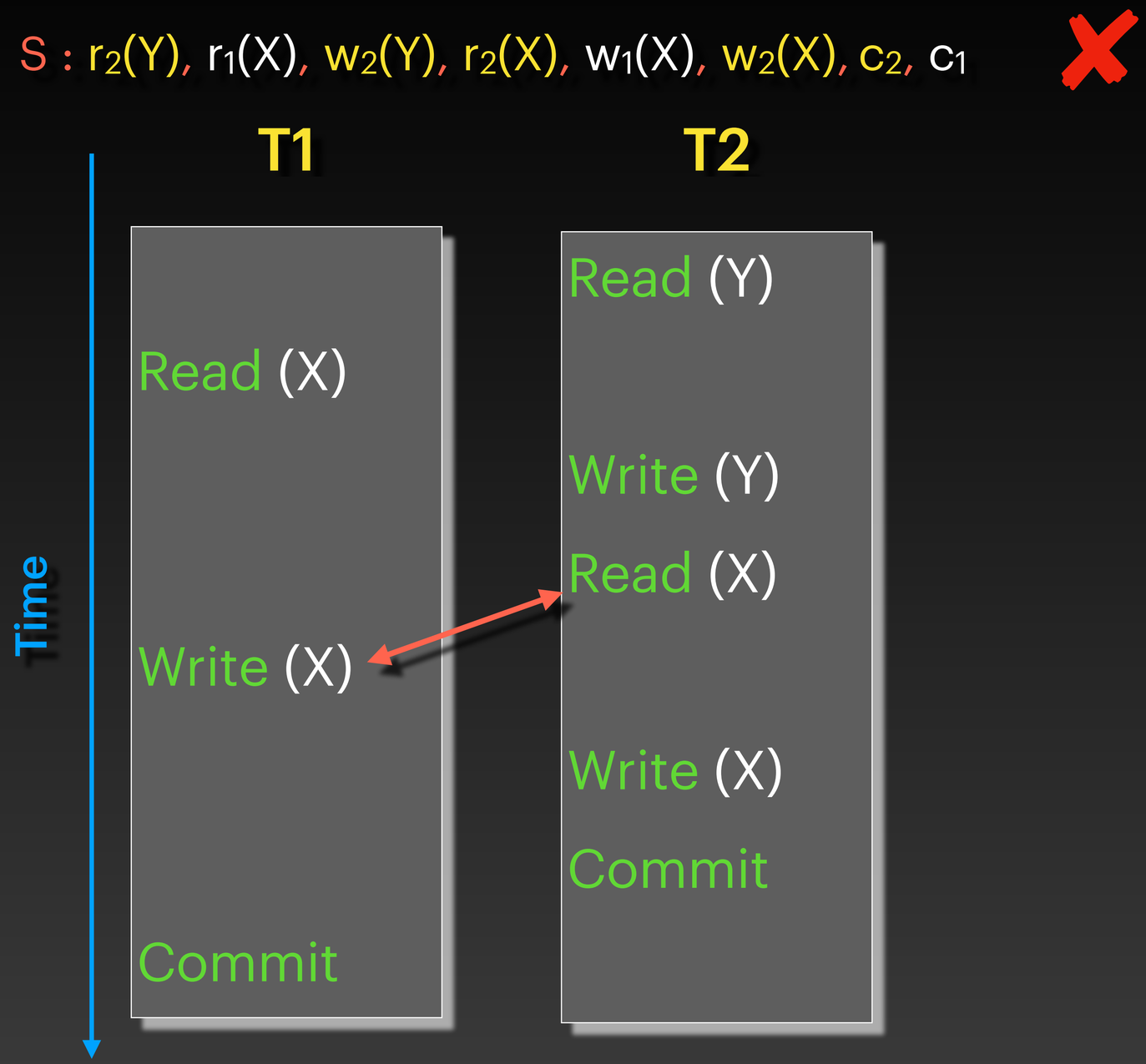
S1 : $r_1(X), w_1(X), c_1, r_2(Y), w_2(Y), r_2(X), w_2(X), c_2$ → Serial

S2 : $r_2(Y), r_1(X), w_2(Y), w_1(X), r_2(X), w_2(X), c_1, c_2$ → Conflict Serializable

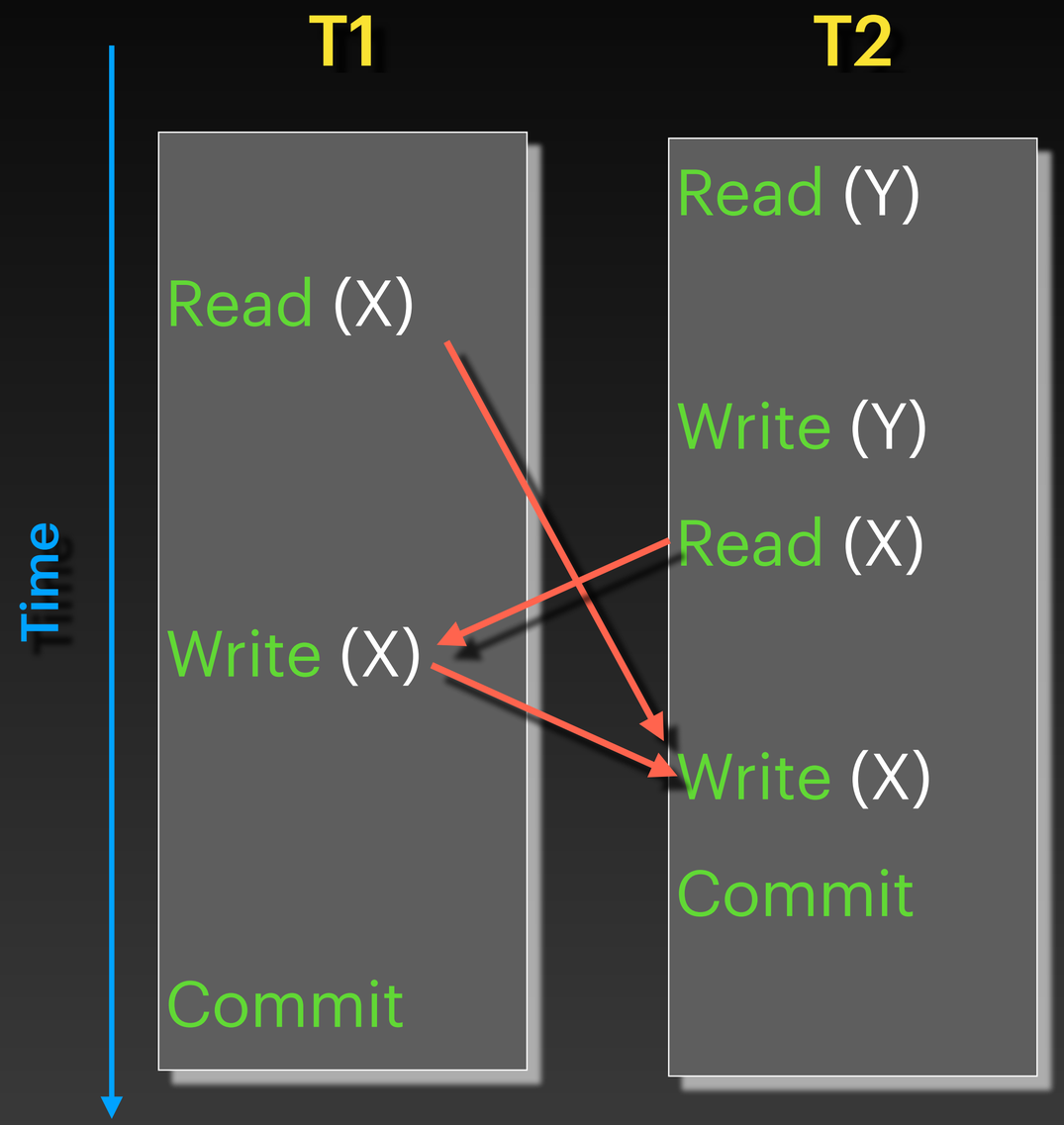
S : $r_2(Y)$, $r_1(X)$, $w_2(Y)$, $w_1(X)$, $r_2(X)$, $w_2(X)$, c_1, c_2 ✓



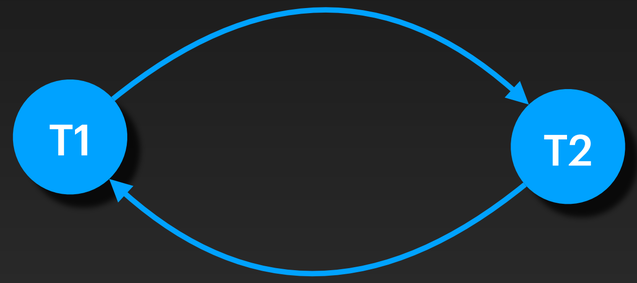
S : $r_2(Y)$, $r_1(X)$, $w_2(Y)$, $r_2(X)$, $w_1(X)$, $w_2(X)$, c_2, c_1



S : $r_2(Y)$, $r_1(X)$, $w_2(Y)$, $r_2(X)$, $w_1(X)$, $w_2(X)$, c_2, c_1

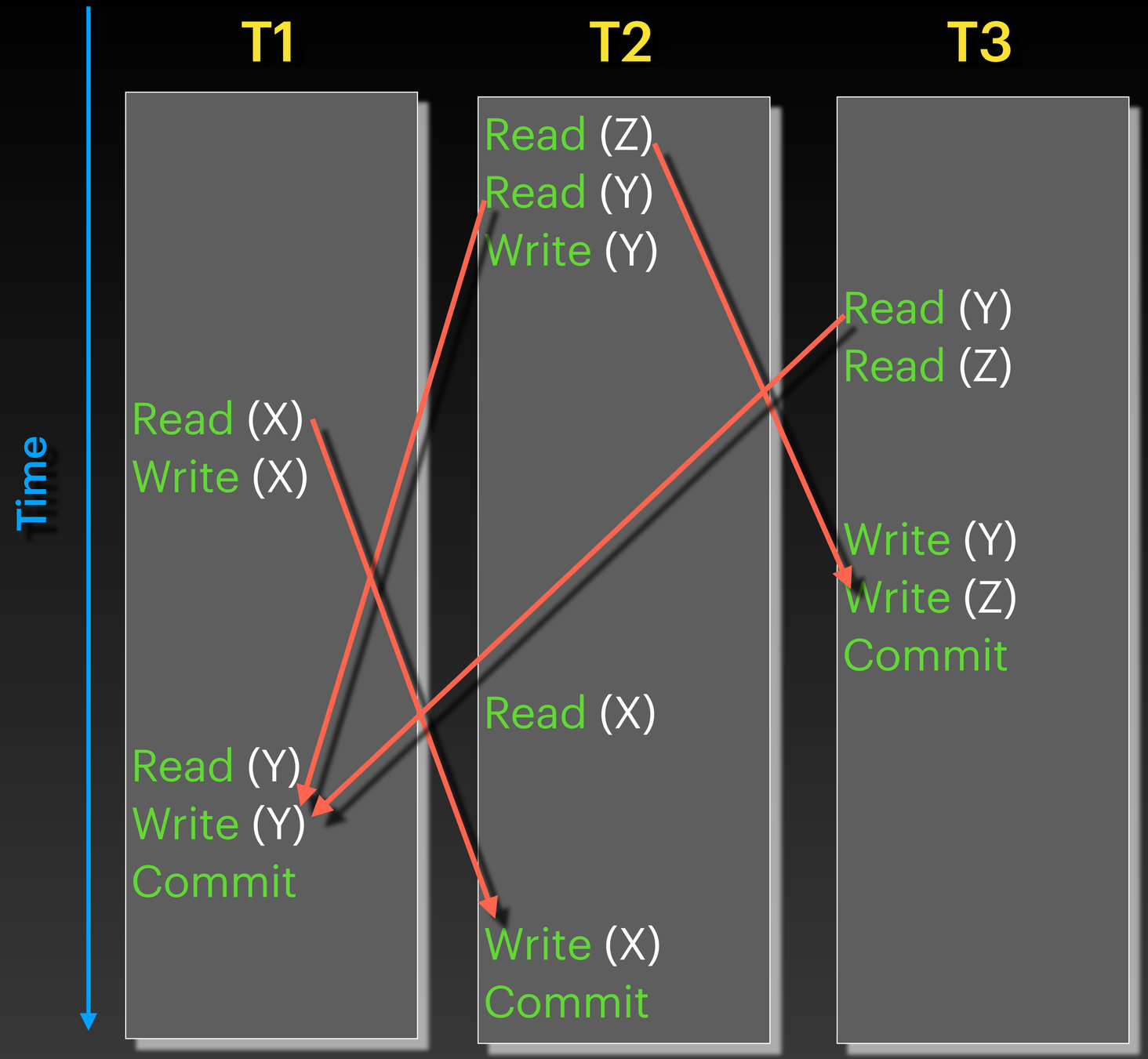


Dependency Graph

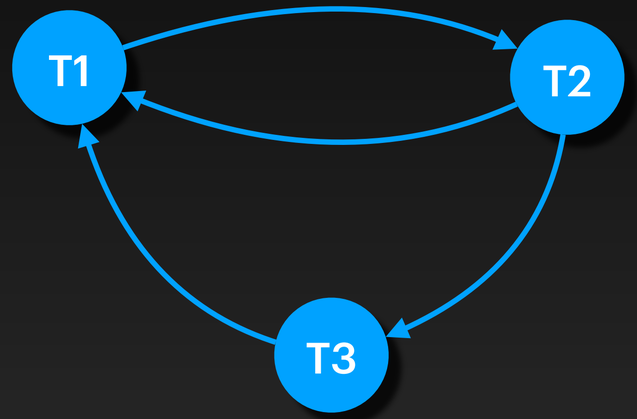


Cycle!!

Not serializable

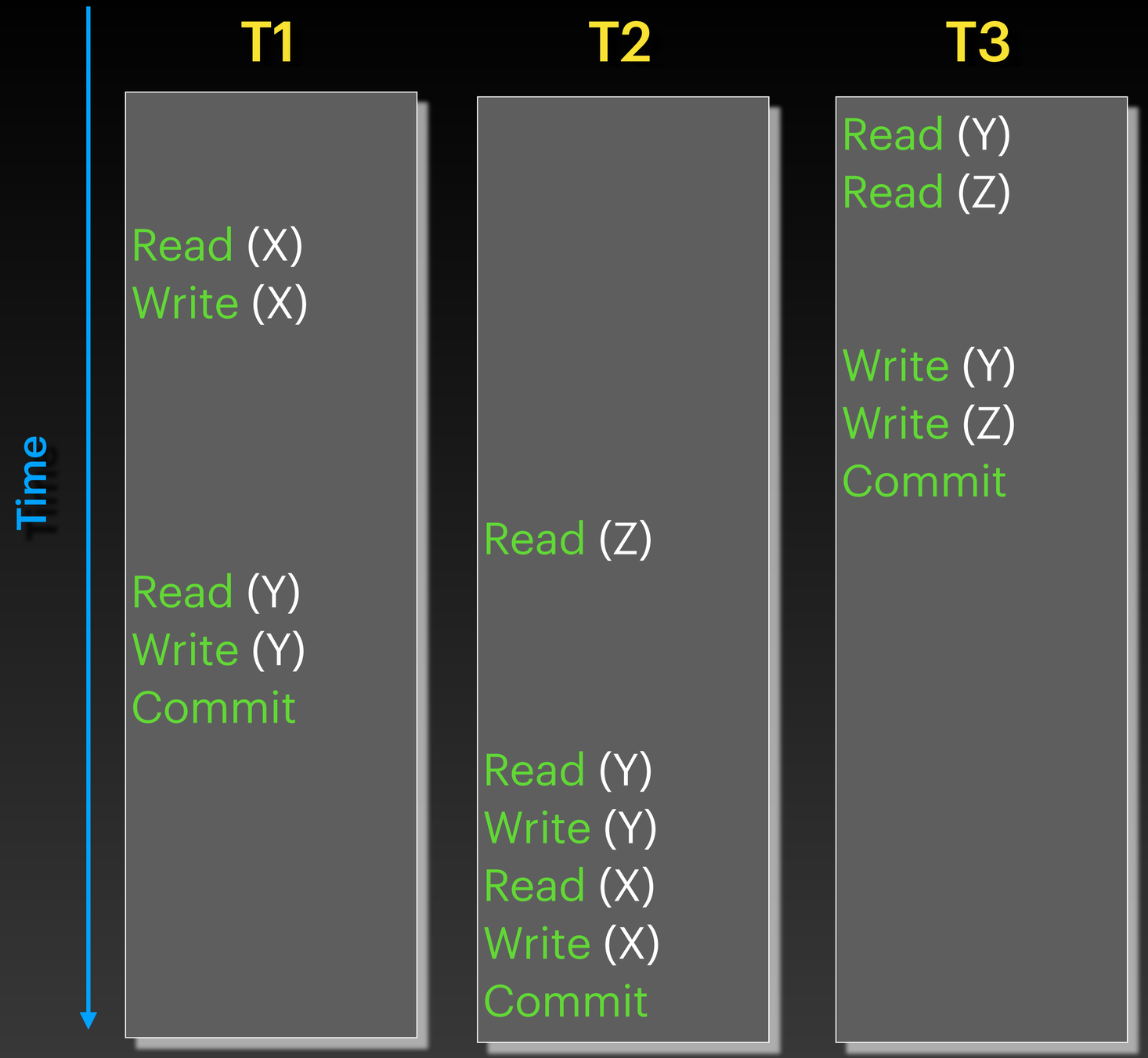


Dependency Graph

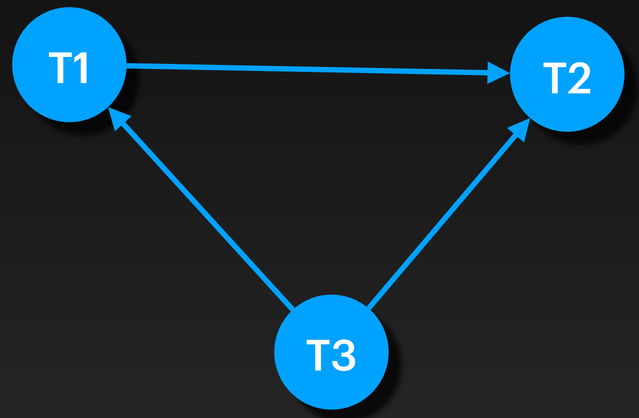


Cycle!!

Not serializable



Dependency Graph



No Cycles!!

Serializable

Equivalent: T3 -> T1 -> T2

Schedule Serializability