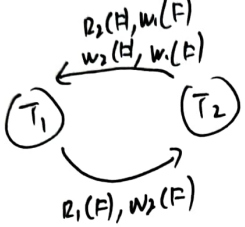
Part A

$R(F)$, $R(P)$, $W(F)$, $W(R)$, C

1. Read committed schedule where all transactions commit that is not conflict serializable

T_1 : $R_1(F)$ $R_1(P)$, $W_1(F)$, $W_1(R)$, C_1

T_2 : $R_2(F)$ $R_2(P)$, $W_2(F)$, $W_2(R)$, C_2



acyclic CSG

T

2. Is it possible for deadlock under RC

WLOG, let T_1 be first transaction to request, $\xrightarrow{\text{acqum}}$ X lock on F

all other transactions blocked when trying to $W_i(F)$ or $R_i(F)$ before $W_i(R)$

until T_1 also $W_1(R)$ and commit before releasing lock.

T_1 will never be blocked by another transaction

F

any transaction that acquire X lock on F will block all other transaction before $W_i(R)$

will not be blocked when $W(R)$

3. Repeatable Read schedule when all transaction commit that is not conflict serializable

In order for all transactions to commit $\Rightarrow$ no deadlock

when any transaction e.g. T_1 acquires S lock for F, all other transaction

cannot acquire S lock since it will block $W_1(F)$ when it tries to upgrade to X lock on F

if T_1 will also be blocked from upgrading X lock on F as T_1 also holding on to S lock on F

deadlock $\Rightarrow$ abort

Hence only way for all transaction to commit is serial execution $\Rightarrow$ always conflict serializable

F

4. deadlock under RR

T_1 : $S_1(F)$ $R_1(F)$ ... $R_1(P)$ $UG_1(F)$ → blocked by T_2

T_2 : $S_2(E)$ $R_2(E)$ - - - - $UG_2(F)$ → blocked by T_1

deadlock

(T)

5. SI schedule where all transaction commit that is not MVSS

for all transaction to commit, no concurrent transaction updating same object, else aborted

all transaction are serial ⇒ MVSS

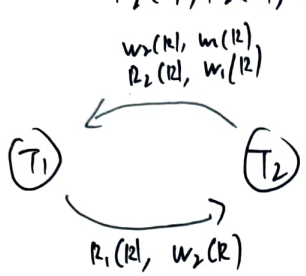
(F)

Part 13 $R(F)$, $R(P)$, $R(R)$, $W(R)$

6. RC schedule when all transactions commit that is not conflict serializable

T_1: $R_1(F), R_1(P), R_1(R)$ $W_1(R), C_1$

T_2: $R_2(F), R_2(P), R_2(R), W_2(R), C_2$



(T)

7. possible deadlock under RC

any transaction that acquires X lock on R will block all other transactions trying to $R_i(R)$ or $W_i(R)$ and will not be blocked until committed

(F)

8. R12 schedule where all transactions commit that is not conflict serializable

For all transaction to commit $\Rightarrow$ no deadlock

when any transaction eg T_1 acquires S lock for R, all other transactions cannot acquire S lock on R since it will block $W_1(R)$ when trying to $UG_1(R)$ itself will also be blocked from $UG_i(R)$ as T_1 holding to S lock on R
deadlock $\Rightarrow$ abort

only way for all transactions to commit is if $R_i(R)$ $W_i(R)$ is executed serially
possible conflicts only go 1 way between transactions $\Rightarrow$ CSG acyclic
always conflict serializable

(T)

9. deadlock under RR

T_1: $R_1(F), R_1(P), S_1(R) R_1(R)$ $\quad UG_1(R)$ → blocked by T_2

T_2: $R_2(F), R_2(P), S_2(R), R_2(R)$ $\quad UG_2(R)$ → blocked by T_1

(T)

IV. If schedule where all transactn commit then it is not MVSS

for all transactn to commit, no concurrent transactn updating R, else aborted

all transactn read ⇒ MVSS

(P)