Theory Assignment 5
CS5280

Darpan Gaur
CO21BTECH11004

Problem 6.3

1st Schedule

Deposit(c) for both transactions t_1 and t_2 are isolated. As shown in figure 1, we can isolate Withdraw(a) and Withdraw(b) by pushing r(q) of Withdraw(a) to right of w(t) of Withdraw(b). Now as all operations at level 1 are isolated we can prune operations at level 0 in the tree.

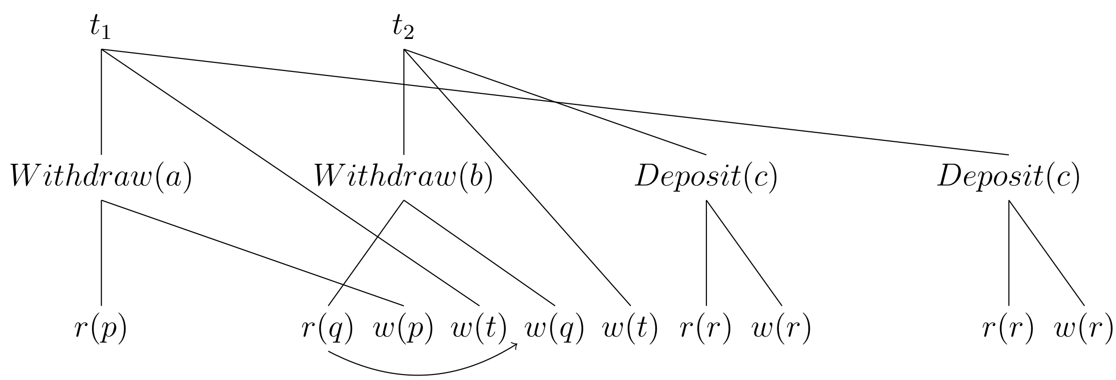

Figure 1: Commute of r(q) in 1st schedule

As shown in figure 2, we can isolate t_1 and t_2 by pushing Deposit(c) of t_1 to the left of Withdraw(b) of t_2. Now as all operations at level 1 are isolated we can prune operations at level 0 in the tree. As we are only left with root nodes, hence 1st schedult is tree reducible. The serialization order is t_1 then t_2.

2nd Schedule

As shown in figure 3, we can isolate Withdraw(a) and Withdraw(b) by pushing r(q) of Withdraw(a) to right of w(t) of Withdraw(b).

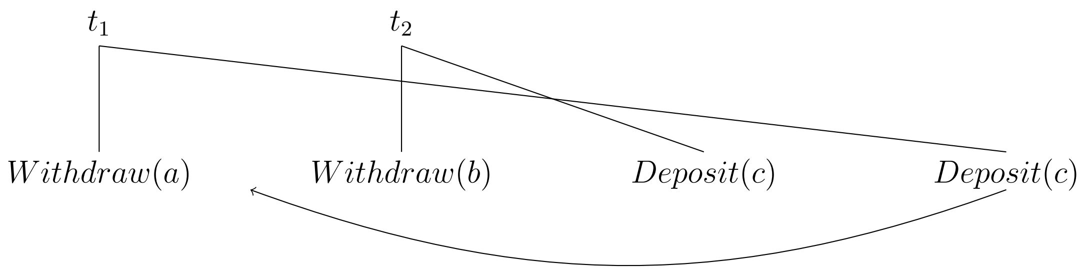

Figure 2: Commute of Deposit(c) in 1st schedule

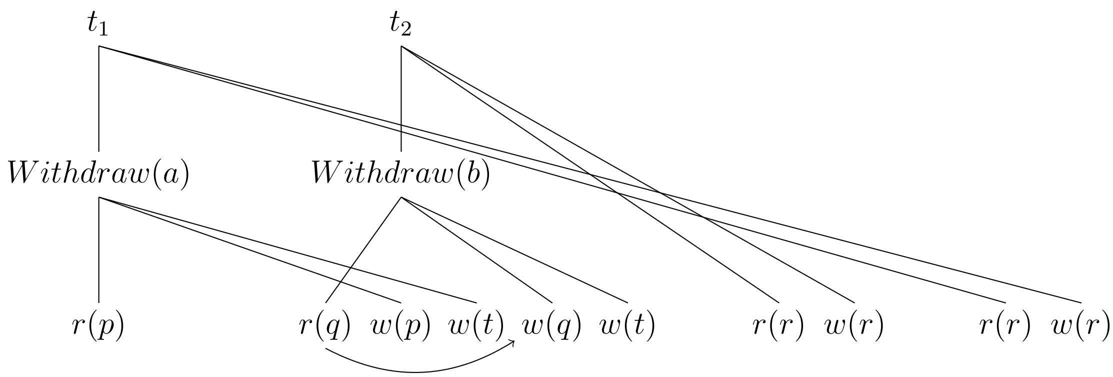

Figure 3: Commute of r(q) in 2nd schedule

As shown in figure 4, we can commute Withdraw(a) and Withdraw(b) as they are operating on different data-items and are isolates. Underlying operations of Withdraw(a) are not conflicting with t_2 r(r) and w(r), we can commute them. Shift Withdraw(a) after t_2. Now as all operations at root are isolated, hence tree is reducible. Execution order is t_2 then t_1.

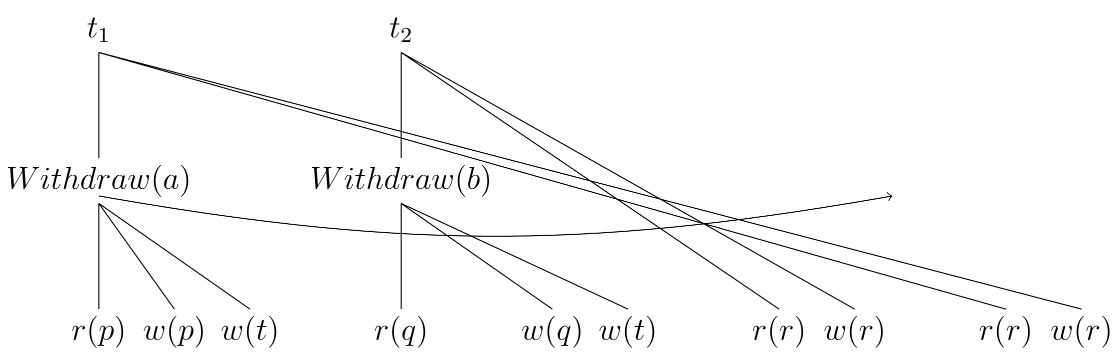

Figure 4: Commute of Withdraw(a) in 2nd schedule

Problem 6.6

Table 1 shows the commutativity of return values of operations for the counter object.

	$Inc \uparrow OK$	$Inc \uparrow No$	$Dec \uparrow OK$	$Dec \uparrow No$	$GV \uparrow x$
$Inc \uparrow OK$	$+$	$-$	$-$	$+$	$-$
$Inc \uparrow No$	$+$	$+$	$-$	$+$	$+$
$Dec \uparrow OK$	$-$	$+$	$+$	$-$	$-$
$Dec \uparrow No$	$-$	$+$	$+$	$+$	$+$
$GV \uparrow x$	$-$	$+$	$-$	$+$	$+$

Table 1: Return value commutativity table

Following improvements of concurrency can be done:

- $dec(x, \Delta_1) \uparrow No$ and $inc(x, \Delta_2) \uparrow OK$: If $x + \Delta_2 - \Delta_1 \leq lower\_bound$, then they are commutable.
- $inc(x, \Delta_1) \uparrow No$ and $dec(x, \Delta_2) \uparrow OK$: If $x + \Delta_1 - \Delta_2 \geq upper\_bound$, then they are commutable.
- $dec(x, \Delta_1) \uparrow OK$ and $inc(x, \Delta_2) \uparrow OK$: If $x + \Delta_2 \leq upper\_bound$ then they are commutable.
- $inc(x, \Delta_1) \uparrow OK$ and $dec(x, \Delta_2) \uparrow OK$: If $x - \Delta_2 \geq lower\_bound$ then they are commutable.
- $dec(x, \Delta_1) \uparrow OK$ and $dec(x, \Delta_2) \uparrow No$: If $x - \Delta_2 \leq lower\_bound$ then they are commutable.
- $inc(x, \Delta_1) \uparrow OK$ and $inc(x, \Delta_2) \uparrow No$: If $x + \Delta_2 \geq upper\_bound$ then they are commutable.

Problem 7.1

Under layered sysytem with layered 2PL, deadlock can occur in two ways:

- Local deadlock: Transactions are wating for each to release the locks due to operations on same layer.
- Global deadlock: Transactions are waiting for each to release the locks due to operations on different layers.

Local Deadlocks

In figure 5, t_1 acquires lock on Withdraw(a), then t_2 acquires lock on Withdraw(b) and t_3 acquires lock on Withdraw(c). Now t_1 is waiting for lock on Deposit(b) which is held by t_2, t_2 is waiting for lock on Deposit(c) which is held by t_3 and t_3 is waiting for lock on Deposit(a) which is held by t_1. This is a deadlock as all transactions are waiting for each other to release the locks. As all are on same layer, hence it is a deadlock within same layer. Here, transactions are blocked at level L_i and if it dosen't hold locks at lower levels, it can't be blocked by any other object below level L_i.

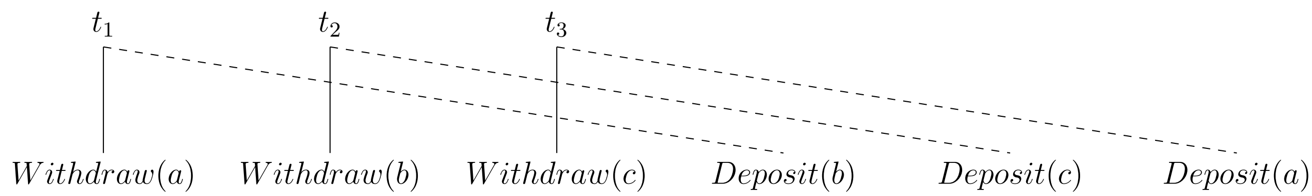

Figure 5: Local deadlock: within same layer

Global Deadlocks

In figure 6, t_1 acquires lock on r(a) and t_2 executes Withdraw(x) and acquires lock on r(a). Now t_2 is waiting for lock on w(a) as it conflicts with r(a) of t_1. t_1 is waiting for lock on Deposit(x) which is conflicting with Withdraw(x) of t_2. This is a deadlock as both transactions are waiting for each other to release the locks. As they are on different layers, hence it is a global deadlock.

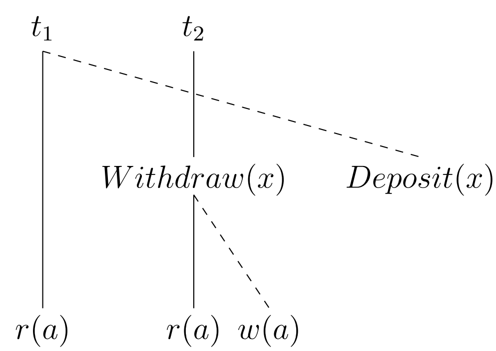

Figure 6: Global deadlock: cross layer