**Tests run examples:**

Note: These output examples use our default printing settings. Except for the data we were asked to print, our logging prints additional information about the state of the structures in the system during the execution. Among other things, it logs the action that is being executed. We want to emphasize that our testing driver allows the user to choose the wanted logging style. Each log type has a dedicated argument that can be set in order to turn these printing on/off. Moreover, there is an option to use the exact print style as explained in the assignment instructions. More about it in our attached documentation pdf.

**Basic deadlock test:** checks the simple case of deadlock between two transactions when running the **RR** version of the ROMV scheduler.

**The test:**  
2 3

U 0 w(x,1) w(y,2) c0;

U 1 a0=r(x) w(y, a0) c1;

U 2 b0=r(y) w(x, b0) c2;

**Output:**

Locks: Transaction 0 acquired write lock for variable `x`.

Transaction 0 [\*U\*] > w(x, 1) w(y, 2) commit

Locks: Transaction 0 acquired write lock for variable `y`.

Transaction 0 [\*U\*] w(x, 1) > w(y, 2) commit

Transaction 0 [\*U\*] w(x, 1) w(y, 2) > commit

Locks: Transaction 0 released write locks for variables `{'x', 'y'}`.

Serialization point. Timestamp: 1

Locks: Transaction 1 acquired read lock for variable `x`.

Transaction 1 [\*U\*] > a0=r(x)=1 w(y, a0) commit

Locks: Transaction 2 acquired read lock for variable `y`.

Transaction 2 [\*U\*] > b0=r(y)=2 w(x, b0) commit

Transaction 1 [\*U\*] WAIT a0=r(x)=1 > w(y, a0) commit

Waiting for locks from transactions: {2}

Locks: Transaction 2 released read locks for variables `{'y'}`.

Transaction 2 [\*U\*] RESET reason: Deadlock cycle found: [(1, 2), (2, 1)]

Locks: Transaction 1 acquired write lock for variable `y`.

Transaction 1 [\*U\*] a0=r(x)=1 > w(y, a0=1) commit

Transaction 2 [\*U\*] (#2) WAIT > b0=r(y) w(x, b0) commit

Waiting for locks from transactions: {1}

Transaction 1 [\*U\*] a0=r(x)=1 w(y, a0=1) > commit

Locks: Transaction 1 released read locks for variables `{'x'}`.

Locks: Transaction 1 released write locks for variables `{'y'}`.

GC: Add GC job because of updater committed variable `y` with version (2) and there is no active reader since previous version (1) of y.

Serialization point. Timestamp: 2

Locks: Transaction 2 acquired read lock for variable `y`.

Transaction 2 [\*U\*] (#2) > b0=r(y)=1 w(x, b0) commit

Locks: Transaction 2 acquired write lock for variable `x`.

Transaction 2 [\*U\*] (#2) b0=r(y)=1 > w(x, b0=1) commit

Transaction 2 [\*U\*] (#2) b0=r(y)=1 w(x, b0=1) > commit

Locks: Transaction 2 released read locks for variables `{'y'}`.

Locks: Transaction 2 released write locks for variables `{'x'}`.

GC: Add GC job because of updater committed variable `x` with version (3) and there is no active reader since previous version (1) of x.

Serialization point. Timestamp: 3

Data in the end of the run:

{'x': [('1', 3)], 'y': [('1', 2)]}

Serialization order:

[0, 1, 2]

**Basic test:** checks the simple case where transactions need to wait for other transactions because of their locks, while readers gets to read freely, when running the **RR** version of the ROMV scheduler.

**The test:**2 6U 0 w(x,1) w(y,2) w(z,3) w(u,4) w(v,5) c0;U 1 a0=r(x) w(y, a0) a1=r(y) w(u, a1) c1;R 2 b0=r(x) b1=r(y) b2=r(z) b3=r(v) c2;U 3 c0=r(v) c1=r(y) w(u, c0) c3;R 4 d0=r(x) d1=r(u) d2=r(v) d3=r(y) c4;U 5 e0=r(z) w(y, e0) c5;

**Output:**

Locks: Transaction 0 acquired write lock for variable `x`.

Transaction 0 [\*U\*] > w(x, 1) w(y, 2) w(z, 3) w(u, 4) w(v, 5) commit

Locks: Transaction 0 acquired write lock for variable `y`.

Transaction 0 [\*U\*] w(x, 1) > w(y, 2) w(z, 3) w(u, 4) w(v, 5) commit

Locks: Transaction 0 acquired write lock for variable `z`.

Transaction 0 [\*U\*] w(x, 1) w(y, 2) > w(z, 3) w(u, 4) w(v, 5) commit

Locks: Transaction 0 acquired write lock for variable `u`.

Transaction 0 [\*U\*] w(x, 1) w(y, 2) w(z, 3) > w(u, 4) w(v, 5) commit

Locks: Transaction 0 acquired write lock for variable `v`.

Transaction 0 [\*U\*] w(x, 1) w(y, 2) w(z, 3) w(u, 4) > w(v, 5) commit

Transaction 0 [\*U\*] w(x, 1) w(y, 2) w(z, 3) w(u, 4) w(v, 5) > commit

Locks: Transaction 0 released write locks for variables `{'v', 'z', 'x', 'y', 'u'}`.

Serialization point. Timestamp: 1

Locks: Transaction 1 acquired read lock for variable `x`.

Transaction 1 [\*U\*] > a0=r(x)=1 w(y, a0) a1=r(y) w(u, a1) commit

Transaction 2 [-R-] > b0=r(x)=1 b1=r(y) b2=r(z) b3=r(v) commit

Serialization point. Timestamp: 2

Locks: Transaction 3 acquired read lock for variable `v`.

Transaction 3 [\*U\*] > c0=r(v)=5 c1=r(y) w(u, c0) commit

Transaction 4 [-R-] > d0=r(x)=1 d1=r(u) d2=r(v) d3=r(y) commit

Serialization point. Timestamp: 3

Locks: Transaction 5 acquired read lock for variable `z`.

Transaction 5 [\*U\*] > e0=r(z)=3 w(y, e0) commit

Locks: Transaction 1 acquired write lock for variable `y`.

Transaction 1 [\*U\*] a0=r(x)=1 > w(y, a0=1) a1=r(y) w(u, a1) commit

Transaction 2 [-R-] b0=r(x)=1 > b1=r(y)=2 b2=r(z) b3=r(v) commit

Transaction 3 [\*U\*] WAIT c0=r(v)=5 > c1=r(y) w(u, c0) commit

Waiting for locks from transactions: {1}

Transaction 4 [-R-] d0=r(x)=1 > d1=r(u)=4 d2=r(v) d3=r(y) commit

Transaction 5 [\*U\*] WAIT e0=r(z)=3 > w(y, e0) commit

Waiting for locks from transactions: {1}

Locks: Transaction 1 acquired read lock for variable `y`.

Transaction 1 [\*U\*] a0=r(x)=1 w(y, a0=1) > a1=r(y)=1 w(u, a1) commit

Transaction 2 [-R-] b0=r(x)=1 b1=r(y)=2 > b2=r(z)=3 b3=r(v) commit

Transaction 3 [\*U\*] WAIT c0=r(v)=5 > c1=r(y) w(u, c0) commit

Waiting for locks from transactions: {1}

Transaction 4 [-R-] d0=r(x)=1 d1=r(u)=4 > d2=r(v)=5 d3=r(y) commit

Transaction 5 [\*U\*] WAIT e0=r(z)=3 > w(y, e0) commit

Waiting for locks from transactions: {1}

Locks: Transaction 1 acquired write lock for variable `u`.

Transaction 1 [\*U\*] a0=r(x)=1 w(y, a0=1) a1=r(y)=1 > w(u, a1=1) commit

Transaction 2 [-R-] b0=r(x)=1 b1=r(y)=2 b2=r(z)=3 > b3=r(v)=5 commit

Transaction 3 [\*U\*] WAIT c0=r(v)=5 > c1=r(y) w(u, c0) commit

Waiting for locks from transactions: {1}

Transaction 4 [-R-] d0=r(x)=1 d1=r(u)=4 d2=r(v)=5 > d3=r(y)=2 commit

Transaction 5 [\*U\*] WAIT e0=r(z)=3 > w(y, e0) commit

Waiting for locks from transactions: {1}

Transaction 1 [\*U\*] a0=r(x)=1 w(y, a0=1) a1=r(y)=1 w(u, a1=1) > commit

Locks: Transaction 1 released read locks for variables `{'x', 'y'}`.

Locks: Transaction 1 released write locks for variables `{'y', 'u'}`.

GC: The just-committed-updater passes the previous version VariableVersion(variable='y', ts=1) of the just-committed-variable to the left older reader (tid=4 ts=3).

GC: The just-committed-updater passes the previous version VariableVersion(variable='u', ts=1) of the just-committed-variable to the left older reader (tid=4 ts=3).

Serialization point. Timestamp: 4

Transaction 2 [-R-] b0=r(x)=1 b1=r(y)=2 b2=r(z)=3 b3=r(v)=5 > commit

Locks: Transaction 3 acquired read lock for variable `y`.

Transaction 3 [\*U\*] c0=r(v)=5 > c1=r(y)=1 w(u, c0) commit

Transaction 4 [-R-] d0=r(x)=1 d1=r(u)=4 d2=r(v)=5 d3=r(y)=2 > commit

GC: The just-committed-reader marks an old version VariableVersion(variable='u', ts=1) under its responsibility for eviction because there is no older reader.

GC: The just-committed-reader marks an old version VariableVersion(variable='y', ts=1) under its responsibility for eviction because there is no older reader.

Transaction 5 [\*U\*] WAIT e0=r(z)=3 > w(y, e0) commit

Waiting for locks from transactions: {3}

Locks: Transaction 3 acquired write lock for variable `u`.

Transaction 3 [\*U\*] c0=r(v)=5 c1=r(y)=1 > w(u, c0=5) commit

Transaction 5 [\*U\*] WAIT e0=r(z)=3 > w(y, e0) commit

Waiting for locks from transactions: {3}

Transaction 3 [\*U\*] c0=r(v)=5 c1=r(y)=1 w(u, c0=5) > commit

Locks: Transaction 3 released read locks for variables `{'v', 'y'}`.

Locks: Transaction 3 released write locks for variables `{'u'}`.

GC: Add GC job because of updater committed variable `u` with version (5) and there is no active reader since previous version (4) of u.

Serialization point. Timestamp: 5

Locks: Transaction 5 acquired write lock for variable `y`.

Transaction 5 [\*U\*] e0=r(z)=3 > w(y, e0=3) commit

Transaction 5 [\*U\*] e0=r(z)=3 w(y, e0=3) > commit

Locks: Transaction 5 released read locks for variables `{'z'}`.

Locks: Transaction 5 released write locks for variables `{'y'}`.

GC: Add GC job because of updater committed variable `y` with version (6) and there is no active reader since previous version (4) of y.

Serialization point. Timestamp: 6

Data in the end of the run:

{'x': [('1', 1)], 'y': [('3', 6)], 'z': [('3', 1)], 'u': [('5', 5)], 'v': [('5', 1)]}

Serialization order:

[0, 2, 4, 1, 3, 5]

**Deadlock test:** a bit more complicated test to check deadlock. More then one deadlock, with waiting transactions and readers in the system, when running the **RR** version of the ROMV scheduler.

**The test:**

2 6

U 0 w(x,3) w(y,6) w(z,8) c0;

U 1 a0=r(x) w(y,18) a1=r(x) w(y,a1) a2=r(z) a3=r(y) c1;

U 2 b0=r(y) w(x,5) b1=r(z) w(x,b1) b2=r(z) b3=r(y) w(y, b2) c2;

U 3 c0=r(z) c1=r(z) w(z,7) c2=r(x) w(y,c2) w(z,c2) c3=r(z) c3;

R 4 d0=r(x) d1=r(z) d2=r(y) d3=r(x) c4;

R 5 e0=r(y) e1=r(x) e2=r(z) e3=r(z) e4=r(x) c5;

**Output:**

Locks: Transaction 0 acquired write lock for variable `x`.

Transaction 0 [\*U\*] > w(x, 3) w(y, 6) w(z, 8) commit

Locks: Transaction 0 acquired write lock for variable `y`.

Transaction 0 [\*U\*] w(x, 3) > w(y, 6) w(z, 8) commit

Locks: Transaction 0 acquired write lock for variable `z`.

Transaction 0 [\*U\*] w(x, 3) w(y, 6) > w(z, 8) commit

Transaction 0 [\*U\*] w(x, 3) w(y, 6) w(z, 8) > commit

Locks: Transaction 0 released write locks for variables `{'x', 'y', 'z'}`.

Serialization point. Timestamp: 1

Locks: Transaction 1 acquired read lock for variable `x`.

Transaction 1 [\*U\*] > a0=r(x)=3 w(y, 18) a1=r(x) w(y, a1) a2=r(z) a3=r(y) commit

Locks: Transaction 2 acquired read lock for variable `y`.

Transaction 2 [\*U\*] > b0=r(y)=6 w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Locks: Transaction 3 acquired read lock for variable `z`.

Transaction 3 [\*U\*] > c0=r(z)=8 c1=r(z) w(z, 7) c2=r(x) w(y, c2) w(z, c2) c3=r(z) commit

Transaction 4 [-R-] > d0=r(x)=3 d1=r(z) d2=r(y) d3=r(x) commit

Serialization point. Timestamp: 2

Transaction 5 [-R-] > e0=r(y)=6 e1=r(x) e2=r(z) e3=r(z) e4=r(x) commit

Serialization point. Timestamp: 3

Transaction 1 [\*U\*] WAIT a0=r(x)=3 > w(y, 18) a1=r(x) w(y, a1) a2=r(z) a3=r(y) commit

Waiting for locks from transactions: {2}

Locks: Transaction 2 released read locks for variables `{'y'}`.

Transaction 2 [\*U\*] RESET reason: Deadlock cycle found: [(1, 2), (2, 1)]

Transaction 3 [\*U\*] c0=r(z)=8 > c1=r(z)=8 w(z, 7) c2=r(x) w(y, c2) w(z, c2) c3=r(z) commit

Transaction 4 [-R-] d0=r(x)=3 > d1=r(z)=8 d2=r(y) d3=r(x) commit

Transaction 5 [-R-] e0=r(y)=6 > e1=r(x)=3 e2=r(z) e3=r(z) e4=r(x) commit

Locks: Transaction 1 acquired write lock for variable `y`.

Transaction 1 [\*U\*] a0=r(x)=3 > w(y, 18) a1=r(x) w(y, a1) a2=r(z) a3=r(y) commit

Transaction 2 [\*U\*] (#2) WAIT > b0=r(y) w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Waiting for locks from transactions: {1}

Locks: Transaction 3 acquired write lock for variable `z`.

Transaction 3 [\*U\*] c0=r(z)=8 c1=r(z)=8 > w(z, 7) c2=r(x) w(y, c2) w(z, c2) c3=r(z) commit

Transaction 4 [-R-] d0=r(x)=3 d1=r(z)=8 > d2=r(y)=6 d3=r(x) commit

Transaction 5 [-R-] e0=r(y)=6 e1=r(x)=3 > e2=r(z)=8 e3=r(z) e4=r(x) commit

Transaction 1 [\*U\*] a0=r(x)=3 w(y, 18) > a1=r(x)=3 w(y, a1) a2=r(z) a3=r(y) commit

Transaction 2 [\*U\*] (#2) WAIT > b0=r(y) w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Waiting for locks from transactions: {1}

Locks: Transaction 3 acquired read lock for variable `x`.

Transaction 3 [\*U\*] c0=r(z)=8 c1=r(z)=8 w(z, 7) > c2=r(x)=3 w(y, c2) w(z, c2) c3=r(z) commit

Transaction 4 [-R-] d0=r(x)=3 d1=r(z)=8 d2=r(y)=6 > d3=r(x)=3 commit

Transaction 5 [-R-] e0=r(y)=6 e1=r(x)=3 e2=r(z)=8 > e3=r(z)=8 e4=r(x) commit

Transaction 1 [\*U\*] a0=r(x)=3 w(y, 18) a1=r(x)=3 > w(y, a1=3) a2=r(z) a3=r(y) commit

Transaction 2 [\*U\*] (#2) WAIT > b0=r(y) w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Waiting for locks from transactions: {1}

Transaction 3 [\*U\*] WAIT c0=r(z)=8 c1=r(z)=8 w(z, 7) c2=r(x)=3 > w(y, c2) w(z, c2) c3=r(z) commit

Waiting for locks from transactions: {1}

Transaction 4 [-R-] d0=r(x)=3 d1=r(z)=8 d2=r(y)=6 d3=r(x)=3 > commit

Transaction 5 [-R-] e0=r(y)=6 e1=r(x)=3 e2=r(z)=8 e3=r(z)=8 > e4=r(x)=3 commit

Locks: Transaction 1 released read locks for variables `{'x'}`.

Locks: Transaction 1 released write locks for variables `{'y'}`.

Transaction 1 [\*U\*] RESET reason: Deadlock cycle found: [(1, 3), (3, 1)]

Locks: Transaction 2 acquired read lock for variable `y`.

Transaction 2 [\*U\*] (#2) > b0=r(y)=6 w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Transaction 3 [\*U\*] WAIT c0=r(z)=8 c1=r(z)=8 w(z, 7) c2=r(x)=3 > w(y, c2) w(z, c2) c3=r(z) commit

Waiting for locks from transactions: {2}

Transaction 5 [-R-] e0=r(y)=6 e1=r(x)=3 e2=r(z)=8 e3=r(z)=8 e4=r(x)=3 > commit

Locks: Transaction 1 acquired read lock for variable `x`.

Transaction 1 [\*U\*] (#2) > a0=r(x)=3 w(y, 18) a1=r(x) w(y, a1) a2=r(z) a3=r(y) commit

Locks: Transaction 2 released read locks for variables `{'y'}`.

Transaction 2 [\*U\*] (#2) RESET reason: Deadlock cycle found: [(2, 3), (3, 2)]

Locks: Transaction 3 acquired write lock for variable `y`.

Transaction 3 [\*U\*] c0=r(z)=8 c1=r(z)=8 w(z, 7) c2=r(x)=3 > w(y, c2=3) w(z, c2) c3=r(z) commit

Transaction 1 [\*U\*] (#2) WAIT a0=r(x)=3 > w(y, 18) a1=r(x) w(y, a1) a2=r(z) a3=r(y) commit

Waiting for locks from transactions: {3}

Transaction 2 [\*U\*] (#3) WAIT > b0=r(y) w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Waiting for locks from transactions: {3}

Transaction 3 [\*U\*] c0=r(z)=8 c1=r(z)=8 w(z, 7) c2=r(x)=3 w(y, c2=3) > w(z, c2=3) c3=r(z) commit

Transaction 1 [\*U\*] (#2) WAIT a0=r(x)=3 > w(y, 18) a1=r(x) w(y, a1) a2=r(z) a3=r(y) commit

Waiting for locks from transactions: {3}

Transaction 2 [\*U\*] (#3) WAIT > b0=r(y) w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Waiting for locks from transactions: {3}

Transaction 3 [\*U\*] c0=r(z)=8 c1=r(z)=8 w(z, 7) c2=r(x)=3 w(y, c2=3) w(z, c2=3) > c3=r(z)=3 commit

Transaction 1 [\*U\*] (#2) WAIT a0=r(x)=3 > w(y, 18) a1=r(x) w(y, a1) a2=r(z) a3=r(y) commit

Waiting for locks from transactions: {3}

Transaction 2 [\*U\*] (#3) WAIT > b0=r(y) w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Waiting for locks from transactions: {3}

Transaction 3 [\*U\*] c0=r(z)=8 c1=r(z)=8 w(z, 7) c2=r(x)=3 w(y, c2=3) w(z, c2=3) c3=r(z)=3 > commit

Locks: Transaction 3 released read locks for variables `{'x', 'z'}`.

Locks: Transaction 3 released write locks for variables `{'y', 'z'}`.

GC: Add GC job because of updater committed variable `z` with version (4) and there is no active reader since previous version (1) of z.

GC: Add GC job because of updater committed variable `y` with version (4) and there is no active reader since previous version (1) of y.

Serialization point. Timestamp: 4

Locks: Transaction 1 acquired write lock for variable `y`.

Transaction 1 [\*U\*] (#2) a0=r(x)=3 > w(y, 18) a1=r(x) w(y, a1) a2=r(z) a3=r(y) commit

Transaction 2 [\*U\*] (#3) WAIT > b0=r(y) w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Waiting for locks from transactions: {1}

Transaction 1 [\*U\*] (#2) a0=r(x)=3 w(y, 18) > a1=r(x)=3 w(y, a1) a2=r(z) a3=r(y) commit

Transaction 2 [\*U\*] (#3) WAIT > b0=r(y) w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Waiting for locks from transactions: {1}

Transaction 1 [\*U\*] (#2) a0=r(x)=3 w(y, 18) a1=r(x)=3 > w(y, a1=3) a2=r(z) a3=r(y) commit

Transaction 2 [\*U\*] (#3) WAIT > b0=r(y) w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Waiting for locks from transactions: {1}

Locks: Transaction 1 acquired read lock for variable `z`.

Transaction 1 [\*U\*] (#2) a0=r(x)=3 w(y, 18) a1=r(x)=3 w(y, a1=3) > a2=r(z)=3 a3=r(y) commit

Transaction 2 [\*U\*] (#3) WAIT > b0=r(y) w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Waiting for locks from transactions: {1}

Locks: Transaction 1 acquired read lock for variable `y`.

Transaction 1 [\*U\*] (#2) a0=r(x)=3 w(y, 18) a1=r(x)=3 w(y, a1=3) a2=r(z)=3 > a3=r(y)=3 commit

Transaction 2 [\*U\*] (#3) WAIT > b0=r(y) w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Waiting for locks from transactions: {1}

Transaction 1 [\*U\*] (#2) a0=r(x)=3 w(y, 18) a1=r(x)=3 w(y, a1=3) a2=r(z)=3 a3=r(y)=3 > commit

Locks: Transaction 1 released read locks for variables `{'x', 'y', 'z'}`.

Locks: Transaction 1 released write locks for variables `{'y'}`.

GC: Add GC job because of updater committed variable `y` with version (5) and there is no active reader since previous version (4) of y.

Serialization point. Timestamp: 5

Locks: Transaction 2 acquired read lock for variable `y`.

Transaction 2 [\*U\*] (#3) > b0=r(y)=3 w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Locks: Transaction 2 acquired write lock for variable `x`.

Transaction 2 [\*U\*] (#3) b0=r(y)=3 > w(x, 5) b1=r(z) w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Locks: Transaction 2 acquired read lock for variable `z`.

Transaction 2 [\*U\*] (#3) b0=r(y)=3 w(x, 5) > b1=r(z)=3 w(x, b1) b2=r(z) b3=r(y) w(y, b2) commit

Transaction 2 [\*U\*] (#3) b0=r(y)=3 w(x, 5) b1=r(z)=3 > w(x, b1=3) b2=r(z) b3=r(y) w(y, b2) commit

Transaction 2 [\*U\*] (#3) b0=r(y)=3 w(x, 5) b1=r(z)=3 w(x, b1=3) > b2=r(z)=3 b3=r(y) w(y, b2) commit

Transaction 2 [\*U\*] (#3) b0=r(y)=3 w(x, 5) b1=r(z)=3 w(x, b1=3) b2=r(z)=3 > b3=r(y)=3 w(y, b2) commit

Locks: Transaction 2 acquired write lock for variable `y`.

Transaction 2 [\*U\*] (#3) b0=r(y)=3 w(x, 5) b1=r(z)=3 w(x, b1=3) b2=r(z)=3 b3=r(y)=3 > w(y, b2=3) commit

Transaction 2 [\*U\*] (#3) b0=r(y)=3 w(x, 5) b1=r(z)=3 w(x, b1=3) b2=r(z)=3 b3=r(y)=3 w(y, b2=3) > commit

Locks: Transaction 2 released read locks for variables `{'y', 'z'}`.

Locks: Transaction 2 released write locks for variables `{'x', 'y'}`.

GC: Add GC job because of updater committed variable `x` with version (6) and there is no active reader since previous version (1) of x.

GC: Add GC job because of updater committed variable `y` with version (6) and there is no active reader since previous version (5) of y.

Serialization point. Timestamp: 6

Data in the end of the run:

{'x': [('3', 6)], 'y': [('3', 6)], 'z': [('3', 4)]}

Serialization order:

[0, 4, 5, 3, 1, 2]

**Lecture test:** the case that was shown in the lecture, where one of the transactions need to wait for the other one to finish. when running the **RR** version of the ROMV scheduler.

**The test:**

2 6

U 0 w(x,1) w(y,2) w(z,3) c0;

R 1 a0=r(x) a1=r(y) c1;

U 2 w(x,1) b0=r(y) w(y,5) c2;

U 3 c0=r(x) w(x,8) c3;

R 4 d0=r(z) d1=r(x) c4;

R 5 e0=r(z) e1=r(x) c5;

**Output:**

Locks: Transaction 0 acquired write lock for variable `x`.

Transaction 0 [\*U\*] > w(x, 1) w(y, 2) w(z, 3) commit

Locks: Transaction 0 acquired write lock for variable `y`.

Transaction 0 [\*U\*] w(x, 1) > w(y, 2) w(z, 3) commit

Locks: Transaction 0 acquired write lock for variable `z`.

Transaction 0 [\*U\*] w(x, 1) w(y, 2) > w(z, 3) commit

Transaction 0 [\*U\*] w(x, 1) w(y, 2) w(z, 3) > commit

Locks: Transaction 0 released write locks for variables `{'x', 'y', 'z'}`.

Serialization point. Timestamp: 1

Transaction 1 [-R-] > a0=r(x)=1 a1=r(y) commit

Serialization point. Timestamp: 2

Locks: Transaction 2 acquired write lock for variable `x`.

Transaction 2 [\*U\*] > w(x, 1) b0=r(y) w(y, 5) commit

Transaction 3 [\*U\*] WAIT > c0=r(x) w(x, 8) commit

Waiting for locks from transactions: {2}

Transaction 4 [-R-] > d0=r(z)=3 d1=r(x) commit

Serialization point. Timestamp: 3

Transaction 5 [-R-] > e0=r(z)=3 e1=r(x) commit

Serialization point. Timestamp: 4

Transaction 1 [-R-] a0=r(x)=1 > a1=r(y)=2 commit

Locks: Transaction 2 acquired read lock for variable `y`.

Transaction 2 [\*U\*] w(x, 1) > b0=r(y)=2 w(y, 5) commit

Transaction 3 [\*U\*] WAIT > c0=r(x) w(x, 8) commit

Waiting for locks from transactions: {2}

Transaction 4 [-R-] d0=r(z)=3 > d1=r(x)=1 commit

Transaction 5 [-R-] e0=r(z)=3 > e1=r(x)=1 commit

Transaction 1 [-R-] a0=r(x)=1 a1=r(y)=2 > commit

Locks: Transaction 2 acquired write lock for variable `y`.

Transaction 2 [\*U\*] w(x, 1) b0=r(y)=2 > w(y, 5) commit

Transaction 3 [\*U\*] WAIT > c0=r(x) w(x, 8) commit

Waiting for locks from transactions: {2}

Transaction 4 [-R-] d0=r(z)=3 d1=r(x)=1 > commit

Transaction 5 [-R-] e0=r(z)=3 e1=r(x)=1 > commit

Transaction 2 [\*U\*] w(x, 1) b0=r(y)=2 w(y, 5) > commit

Locks: Transaction 2 released read locks for variables `{'y'}`.

Locks: Transaction 2 released write locks for variables `{'x', 'y'}`.

GC: Add GC job because of updater committed variable `x` with version (5) and there is no active reader since previous version (1) of x.

GC: Add GC job because of updater committed variable `y` with version (5) and there is no active reader since previous version (1) of y.

Serialization point. Timestamp: 5

Locks: Transaction 3 acquired read lock for variable `x`.

Transaction 3 [\*U\*] > c0=r(x)=1 w(x, 8) commit

Locks: Transaction 3 acquired write lock for variable `x`.

Transaction 3 [\*U\*] c0=r(x)=1 > w(x, 8) commit

Transaction 3 [\*U\*] c0=r(x)=1 w(x, 8) > commit

Locks: Transaction 3 released read locks for variables `{'x'}`.

Locks: Transaction 3 released write locks for variables `{'x'}`.

GC: Add GC job because of updater committed variable `x` with version (6) and there is no active reader since previous version (5) of x.

Serialization point. Timestamp: 6

Data in the end of the run:

{'x': [('8', 6)], 'y': [('5', 5)], 'z': [('3', 1)]}

Serialization order:

[0, 1, 4, 5, 2, 3]

**RR version test:** check the behavior of long readers (used to show when versions are being deleted), when running the **RR** version of the ROMV scheduler.

\*the GC output is also being shown in here

**The test:**

2 7

U 0 w(x,3) w(y,7) w(z,10) c0;

R 1 a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) c1;

U 2 w(x,19) c2;

U 3 w(y,18) b0=r(y) w(y,17) c3;

R 4 c0=r(z) c1=r(z) c2=r(y) c3=r(z) c4;

U 5 w(z,17) w(z,9) w(y,9) c5;

R 6 d0=r(x) d0=r(x) d0=r(x) d1=r(z) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d2=r(y) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) c6;

**Output:**

Locks: Transaction 0 acquired write lock for variable `x`.

Transaction 0 [\*U\*] > w(x, 3) w(y, 7) w(z, 10) commit

Locks: Transaction 0 acquired write lock for variable `y`.

Transaction 0 [\*U\*] w(x, 3) > w(y, 7) w(z, 10) commit

Locks: Transaction 0 acquired write lock for variable `z`.

Transaction 0 [\*U\*] w(x, 3) w(y, 7) > w(z, 10) commit

Transaction 0 [\*U\*] w(x, 3) w(y, 7) w(z, 10) > commit

Locks: Transaction 0 released write locks for variables `{'x', 'y', 'z'}`.

Serialization point. Timestamp: 1

Transaction 1 [-R-] > a0=r(x)=3 a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) commit

Serialization point. Timestamp: 2

Locks: Transaction 2 acquired write lock for variable `x`.

Transaction 2 [\*U\*] > w(x, 19) commit

Locks: Transaction 3 acquired write lock for variable `y`.

Transaction 3 [\*U\*] > w(y, 18) b0=r(y) w(y, 17) commit

Transaction 4 [-R-] > c0=r(z)=10 c1=r(z) c2=r(y) c3=r(z) commit

Serialization point. Timestamp: 3

Locks: Transaction 5 acquired write lock for variable `z`.

Transaction 5 [\*U\*] > w(z, 17) w(z, 9) w(y, 9) commit

Transaction 6 [-R-] > d0=r(x)=3 d0=r(x) d0=r(x) d1=r(z) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d2=r(y) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Serialization point. Timestamp: 4

Transaction 1 [-R-] a0=r(x)=3 > a0=r(x)=3 a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) commit

Transaction 2 [\*U\*] w(x, 19) > commit

Locks: Transaction 2 released write locks for variables `{'x'}`.

GC: The just-committed-updater passes the previous version VariableVersion(variable='x', ts=1) of the just-committed-variable to the left older reader (tid=6 ts=4).

Serialization point. Timestamp: 5

Locks: Transaction 3 acquired read lock for variable `y`.

Transaction 3 [\*U\*] w(y, 18) > b0=r(y)=18 w(y, 17) commit

Transaction 4 [-R-] c0=r(z)=10 > c1=r(z)=10 c2=r(y) c3=r(z) commit

Transaction 5 [\*U\*] w(z, 17) > w(z, 9) w(y, 9) commit

Transaction 6 [-R-] d0=r(x)=3 > d0=r(x)=3 d0=r(x) d1=r(z) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d2=r(y) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 1 [-R-] a0=r(x)=3 a0=r(x)=3 > a0=r(x)=3 a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) commit

Transaction 3 [\*U\*] w(y, 18) b0=r(y)=18 > w(y, 17) commit

Transaction 4 [-R-] c0=r(z)=10 c1=r(z)=10 > c2=r(y)=7 c3=r(z) commit

Transaction 5 [\*U\*] WAIT w(z, 17) w(z, 9) > w(y, 9) commit

Waiting for locks from transactions: {3}

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 d1=r(z) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d2=r(y) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 1 [-R-] a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 > a0=r(x)=3 a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) commit

Transaction 3 [\*U\*] w(y, 18) b0=r(y)=18 w(y, 17) > commit

Locks: Transaction 3 released read locks for variables `{'y'}`.

Locks: Transaction 3 released write locks for variables `{'y'}`.

GC: The just-committed-updater passes the previous version VariableVersion(variable='y', ts=1) of the just-committed-variable to the left older reader (tid=6 ts=4).

Serialization point. Timestamp: 6

Transaction 4 [-R-] c0=r(z)=10 c1=r(z)=10 c2=r(y)=7 > c3=r(z)=10 commit

Locks: Transaction 5 acquired write lock for variable `y`.

Transaction 5 [\*U\*] w(z, 17) w(z, 9) > w(y, 9) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > d1=r(z)=10 d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d2=r(y) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 1 [-R-] a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 > a0=r(x)=3 a0=r(x) a0=r(x) a0=r(x) a0=r(x) a0=r(x) commit

Transaction 4 [-R-] c0=r(z)=10 c1=r(z)=10 c2=r(y)=7 c3=r(z)=10 > commit

Transaction 5 [\*U\*] w(z, 17) w(z, 9) w(y, 9) > commit

Locks: Transaction 5 released write locks for variables `{'y', 'z'}`.

GC: The just-committed-updater passes the previous version VariableVersion(variable='z', ts=1) of the just-committed-variable to the left older reader (tid=6 ts=4).

GC: Add GC job because of updater committed variable `y` with version (7) and there is no active reader since previous version (6) of y.

Serialization point. Timestamp: 7

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 > d0=r(x)=3 d0=r(x) d0=r(x) d0=r(x) d0=r(x) d2=r(y) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 1 [-R-] a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 > a0=r(x)=3 a0=r(x) a0=r(x) a0=r(x) a0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 > d0=r(x)=3 d0=r(x) d0=r(x) d0=r(x) d2=r(y) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 1 [-R-] a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 > a0=r(x)=3 a0=r(x) a0=r(x) a0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 d0=r(x) d0=r(x) d2=r(y) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 1 [-R-] a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 > a0=r(x)=3 a0=r(x) a0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 d0=r(x) d2=r(y) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 1 [-R-] a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 > a0=r(x)=3 a0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 d2=r(y) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 1 [-R-] a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 > a0=r(x)=3 commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > d2=r(y)=7 d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 1 [-R-] a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 a0=r(x)=3 > commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d2=r(y)=7 > d0=r(x)=3 d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d2=r(y)=7 d0=r(x)=3 > d0=r(x)=3 d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d2=r(y)=7 d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d2=r(y)=7 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d2=r(y)=7 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 d0=r(x) d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d2=r(y)=7 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 d0=r(x) d0=r(x) d0=r(x) d0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d2=r(y)=7 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 d0=r(x) d0=r(x) d0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d2=r(y)=7 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 d0=r(x) d0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d2=r(y)=7 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 d0=r(x) commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d2=r(y)=7 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > d0=r(x)=3 commit

Transaction 6 [-R-] d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d1=r(z)=10 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d2=r(y)=7 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 d0=r(x)=3 > commit

GC: The just-committed-reader marks an old version VariableVersion(variable='z', ts=1) under its responsibility for eviction because there is no older reader.

GC: The just-committed-reader marks an old version VariableVersion(variable='y', ts=1) under its responsibility for eviction because there is no older reader.

GC: The just-committed-reader marks an old version VariableVersion(variable='x', ts=1) under its responsibility for eviction because there is no older reader.

Data in the end of the run:

{'x': [('19', 5)], 'y': [('9', 7)], 'z': [('9', 7)]}

Serialization order:

[0, 1, 4, 6, 2, 3, 5]

**Serial versions test:** checks the versions handling of a serial run. running the **serial** version of the ROMV scheduler  
\*the GC output is also being shown in here

**The test:**

1 9

U 0 w(x,3) w(y,7) w(z,10) c0;

R 1 a0=r(x) c1;

U 2 b2=r(z) w(x,b2) c2;

R 3 c0=r(z) c1=r(z) c3;

U 4 w(x,11) d2=r(y) d3=r(x) c4;

R 5 e0=r(y) e1=r(x) e2=r(z) e3=r(z) e4=r(x) c5;

U 6 w(x,13) w(z,14) c6;

U 7 w(x,15) w(y,15) c7;

R 8 b1=r(x) b2=r(z) c8;

**Output:**

Locks: Transaction 0 acquired write lock for variable `x`.

Transaction 0 [\*U\*] > w(x, 3) w(y, 7) w(z, 10) commit

Locks: Transaction 0 acquired write lock for variable `y`.

Transaction 0 [\*U\*] w(x, 3) > w(y, 7) w(z, 10) commit

Locks: Transaction 0 acquired write lock for variable `z`.

Transaction 0 [\*U\*] w(x, 3) w(y, 7) > w(z, 10) commit

Transaction 0 [\*U\*] w(x, 3) w(y, 7) w(z, 10) > commit

Locks: Transaction 0 released write locks for variables `{'x', 'y', 'z'}`.

Serialization point. Timestamp: 1

Transaction 1 [-R-] > a0=r(x)=3 commit

Serialization point. Timestamp: 2

Transaction 1 [-R-] a0=r(x)=3 > commit

Locks: Transaction 2 acquired read lock for variable `z`.

Transaction 2 [\*U\*] > b2=r(z)=10 w(x, b2) commit

Locks: Transaction 2 acquired write lock for variable `x`.

Transaction 2 [\*U\*] b2=r(z)=10 > w(x, b2=10) commit

Transaction 2 [\*U\*] b2=r(z)=10 w(x, b2=10) > commit

Locks: Transaction 2 released read locks for variables `{'z'}`.

Locks: Transaction 2 released write locks for variables `{'x'}`.

GC: Add GC job because of updater committed variable `x` with version (3) and there is no active reader since previous version (1) of x.

Serialization point. Timestamp: 3

Transaction 3 [-R-] > c0=r(z)=10 c1=r(z) commit

Serialization point. Timestamp: 4

Transaction 3 [-R-] c0=r(z)=10 > c1=r(z)=10 commit

Transaction 3 [-R-] c0=r(z)=10 c1=r(z)=10 > commit

Locks: Transaction 4 acquired write lock for variable `x`.

Transaction 4 [\*U\*] > w(x, 11) d2=r(y) d3=r(x) commit

Locks: Transaction 4 acquired read lock for variable `y`.

Transaction 4 [\*U\*] w(x, 11) > d2=r(y)=7 d3=r(x) commit

Locks: Transaction 4 acquired read lock for variable `x`.

Transaction 4 [\*U\*] w(x, 11) d2=r(y)=7 > d3=r(x)=11 commit

Transaction 4 [\*U\*] w(x, 11) d2=r(y)=7 d3=r(x)=11 > commit

Locks: Transaction 4 released read locks for variables `{'x', 'y'}`.

Locks: Transaction 4 released write locks for variables `{'x'}`.

GC: Add GC job because of updater committed variable `x` with version (5) and there is no active reader since previous version (3) of x.

Serialization point. Timestamp: 5

Transaction 5 [-R-] > e0=r(y)=7 e1=r(x) e2=r(z) e3=r(z) e4=r(x) commit

Serialization point. Timestamp: 6

Transaction 5 [-R-] e0=r(y)=7 > e1=r(x)=11 e2=r(z) e3=r(z) e4=r(x) commit

Transaction 5 [-R-] e0=r(y)=7 e1=r(x)=11 > e2=r(z)=10 e3=r(z) e4=r(x) commit

Transaction 5 [-R-] e0=r(y)=7 e1=r(x)=11 e2=r(z)=10 > e3=r(z)=10 e4=r(x) commit

Transaction 5 [-R-] e0=r(y)=7 e1=r(x)=11 e2=r(z)=10 e3=r(z)=10 > e4=r(x)=11 commit

Transaction 5 [-R-] e0=r(y)=7 e1=r(x)=11 e2=r(z)=10 e3=r(z)=10 e4=r(x)=11 > commit

Locks: Transaction 6 acquired write lock for variable `x`.

Transaction 6 [\*U\*] > w(x, 13) w(z, 14) commit

Locks: Transaction 6 acquired write lock for variable `z`.

Transaction 6 [\*U\*] w(x, 13) > w(z, 14) commit

Transaction 6 [\*U\*] w(x, 13) w(z, 14) > commit

Locks: Transaction 6 released write locks for variables `{'x', 'z'}`.

GC: Add GC job because of updater committed variable `x` with version (7) and there is no active reader since previous version (5) of x.

GC: Add GC job because of updater committed variable `z` with version (7) and there is no active reader since previous version (1) of z.

Serialization point. Timestamp: 7

Locks: Transaction 7 acquired write lock for variable `x`.

Transaction 7 [\*U\*] > w(x, 15) w(y, 15) commit

Locks: Transaction 7 acquired write lock for variable `y`.

Transaction 7 [\*U\*] w(x, 15) > w(y, 15) commit

Transaction 7 [\*U\*] w(x, 15) w(y, 15) > commit

Locks: Transaction 7 released write locks for variables `{'x', 'y'}`.

GC: Add GC job because of updater committed variable `x` with version (8) and there is no active reader since previous version (7) of x.

GC: Add GC job because of updater committed variable `y` with version (8) and there is no active reader since previous version (1) of y.

Serialization point. Timestamp: 8

Transaction 8 [-R-] > b1=r(x)=15 b2=r(z) commit

Serialization point. Timestamp: 9

Transaction 8 [-R-] b1=r(x)=15 > b2=r(z)=14 commit

Transaction 8 [-R-] b1=r(x)=15 b2=r(z)=14 > commit

Data in the end of the run:

{'x': [('15', 8)], 'y': [('15', 8)], 'z': [('14', 7)]}

**Suspend test:** Checks the behavior of the scheduler when readers get into the system in different times (including - after an updater has finished his commit, and before it), when running the **RR** version of the ROMV scheduler.

Note: We use here a dedicated operation "suspend" that allows us to suspend the entrance of the reader to the system. The transaction yields for this epoch and lets the following transaction continue right after.

**The test:**

2 7

U 0 w(x,1) w(y,2) w(z,3) w(u,4) w(v,5) c0;

R 1 a0=r(x) a2=r(v) a3=r(u) c1;

U 2 c0=r(v) w(u, c0) c2;

U 3 d0=r(z) d1=r(y) w(y, d0) d2=r(y) c3;

R 4 suspend suspend suspend f3=r(y) f4=r(x) c4;

R 5 suspend suspend suspend suspend e0=r(y) e1=r(x) c5;

U 6 s0=r(x) w(x, 8) c6;

**Output:**

Locks: Transaction 0 acquired write lock for variable `x`.

Transaction 0 [\*U\*] > w(x, 1) w(y, 2) w(z, 3) w(u, 4) w(v, 5) commit

Locks: Transaction 0 acquired write lock for variable `y`.

Transaction 0 [\*U\*] w(x, 1) > w(y, 2) w(z, 3) w(u, 4) w(v, 5) commit

Locks: Transaction 0 acquired write lock for variable `z`.

Transaction 0 [\*U\*] w(x, 1) w(y, 2) > w(z, 3) w(u, 4) w(v, 5) commit

Locks: Transaction 0 acquired write lock for variable `u`.

Transaction 0 [\*U\*] w(x, 1) w(y, 2) w(z, 3) > w(u, 4) w(v, 5) commit

Locks: Transaction 0 acquired write lock for variable `v`.

Transaction 0 [\*U\*] w(x, 1) w(y, 2) w(z, 3) w(u, 4) > w(v, 5) commit

Transaction 0 [\*U\*] w(x, 1) w(y, 2) w(z, 3) w(u, 4) w(v, 5) > commit

Locks: Transaction 0 released write locks for variables `{'v', 'z', 'x', 'y', 'u'}`.

Serialization point. Timestamp: 1

Transaction 1 [-R-] > a0=r(x)=1 a2=r(v) a3=r(u) commit

Serialization point. Timestamp: 2

Locks: Transaction 2 acquired read lock for variable `v`.

Transaction 2 [\*U\*] > c0=r(v)=5 w(u, c0) commit

Locks: Transaction 3 acquired read lock for variable `z`.

Transaction 3 [\*U\*] > d0=r(z)=3 d1=r(y) w(y, d0) d2=r(y) commit

Transaction 4 [-R-] > suspend suspend suspend f3=r(y) f4=r(x) commit

Transaction 5 [-R-] > suspend suspend suspend suspend e0=r(y) e1=r(x) commit

Locks: Transaction 6 acquired read lock for variable `x`.

Transaction 6 [\*U\*] > s0=r(x)=1 w(x, 8) commit

Transaction 1 [-R-] a0=r(x)=1 > a2=r(v)=5 a3=r(u) commit

Locks: Transaction 2 acquired write lock for variable `u`.

Transaction 2 [\*U\*] c0=r(v)=5 > w(u, c0=5) commit

Locks: Transaction 3 acquired read lock for variable `y`.

Transaction 3 [\*U\*] d0=r(z)=3 > d1=r(y)=2 w(y, d0) d2=r(y) commit

Transaction 4 [-R-] suspend > suspend suspend f3=r(y) f4=r(x) commit

Transaction 5 [-R-] suspend > suspend suspend suspend e0=r(y) e1=r(x) commit

Locks: Transaction 6 acquired write lock for variable `x`.

Transaction 6 [\*U\*] s0=r(x)=1 > w(x, 8) commit

Transaction 1 [-R-] a0=r(x)=1 a2=r(v)=5 > a3=r(u)=4 commit

Transaction 2 [\*U\*] c0=r(v)=5 w(u, c0=5) > commit

Locks: Transaction 2 released read locks for variables `{'v'}`.

Locks: Transaction 2 released write locks for variables `{'u'}`.

GC: The just-committed-updater passes the previous version VariableVersion(variable='u', ts=1) of the just-committed-variable to the left older reader (tid=1 ts=2).

Serialization point. Timestamp: 3

Locks: Transaction 3 acquired write lock for variable `y`.

Transaction 3 [\*U\*] d0=r(z)=3 d1=r(y)=2 > w(y, d0=3) d2=r(y) commit

Transaction 4 [-R-] suspend suspend > suspend f3=r(y) f4=r(x) commit

Transaction 5 [-R-] suspend suspend > suspend suspend e0=r(y) e1=r(x) commit

Transaction 6 [\*U\*] s0=r(x)=1 w(x, 8) > commit

Locks: Transaction 6 released read locks for variables `{'x'}`.

Locks: Transaction 6 released write locks for variables `{'x'}`.

GC: The just-committed-updater passes the previous version VariableVersion(variable='x', ts=1) of the just-committed-variable to the left older reader (tid=1 ts=2).

Serialization point. Timestamp: 4

Transaction 1 [-R-] a0=r(x)=1 a2=r(v)=5 a3=r(u)=4 > commit

GC: The just-committed-reader marks an old version VariableVersion(variable='u', ts=1) under its responsibility for eviction because there is no older reader.

GC: The just-committed-reader marks an old version VariableVersion(variable='x', ts=1) under its responsibility for eviction because there is no older reader.

Transaction 3 [\*U\*] d0=r(z)=3 d1=r(y)=2 w(y, d0=3) > d2=r(y)=3 commit

Transaction 4 [-R-] suspend suspend suspend > f3=r(y)=2 f4=r(x) commit

Serialization point. Timestamp: 5

Transaction 5 [-R-] suspend suspend suspend > suspend e0=r(y) e1=r(x) commit

Transaction 3 [\*U\*] d0=r(z)=3 d1=r(y)=2 w(y, d0=3) d2=r(y)=3 > commit

Locks: Transaction 3 released read locks for variables `{'y', 'z'}`.

Locks: Transaction 3 released write locks for variables `{'y'}`.

GC: The just-committed-updater passes the previous version VariableVersion(variable='y', ts=1) of the just-committed-variable to the left older reader (tid=4 ts=5).

Serialization point. Timestamp: 6

Transaction 4 [-R-] suspend suspend suspend f3=r(y)=2 > f4=r(x)=8 commit

Transaction 5 [-R-] suspend suspend suspend suspend > e0=r(y)=3 e1=r(x) commit

Serialization point. Timestamp: 7

Transaction 4 [-R-] suspend suspend suspend f3=r(y)=2 f4=r(x)=8 > commit

GC: The just-committed-reader marks an old version VariableVersion(variable='y', ts=1) under its responsibility for eviction because there is no older reader.

Transaction 5 [-R-] suspend suspend suspend suspend e0=r(y)=3 > e1=r(x)=8 commit

Transaction 5 [-R-] suspend suspend suspend suspend e0=r(y)=3 e1=r(x)=8 > commit

Data in the end of the run:

{'x': [('8', 4)], 'y': [('3', 6)], 'z': [('3', 1)], 'u': [('5', 3)], 'v': [('5', 1)]}

Serialization order:

[0, 1, 2, 6, 4, 3, 5]