**Tests run examples:**(this output example uses the special printing that shows the action that is being executed)

**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:**

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

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

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

ROMV RR | Serialization point. Timestamp: 1

ROMV RR |

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

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

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

ROMV RR | Waiting for locks from transactions: {2}

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

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

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

ROMV RR | Waiting for locks from transactions: {1}

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

ROMV RR | Serialization point. Timestamp: 2

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

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

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

ROMV RR | Serialization point. Timestamp: 3

ROMV RR | Data in the end of the run:

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

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

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

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

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

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

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

ROMV RR | Serialization point. Timestamp: 1

ROMV RR |

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

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

ROMV RR | Serialization point. Timestamp: 2

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

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

ROMV RR | Serialization point. Timestamp: 3

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

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

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

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

ROMV RR | Waiting for locks from transactions: {1}

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

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

ROMV RR | Waiting for locks from transactions: {1}

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

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

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

ROMV RR | Waiting for locks from transactions: {1}

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

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

ROMV RR | Waiting for locks from transactions: {1}

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

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

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

ROMV RR | Waiting for locks from transactions: {1}

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

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

ROMV RR | Waiting for locks from transactions: {1}

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

ROMV RR | Serialization point. Timestamp: 4

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

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

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

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

ROMV RR | Waiting for locks from transactions: {3}

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

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

ROMV RR | Waiting for locks from transactions: {3}

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

ROMV RR | Serialization point. Timestamp: 5

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

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

ROMV RR | Serialization point. Timestamp: 6

ROMV RR |

ROMV RR |

ROMV RR | Data in the end of the run:

ROMV RR | {'y': [('3', 6)], 'v': [('5', 1)], 'z': [('3', 1)], 'x': [('1', 1)], 'u': [('5', 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:**

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

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

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

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

ROMV RR | Serialization point. Timestamp: 1

ROMV RR |

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

ROMV RR | 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

ROMV RR | 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

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

ROMV RR | Serialization point. Timestamp: 2

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

ROMV RR | Serialization point. Timestamp: 3

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

ROMV RR | Waiting for locks from transactions: {2}

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

ROMV RR | 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

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

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

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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {1}

ROMV RR | 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

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

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

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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {1}

ROMV RR | 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

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

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

ROMV RR | 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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {1}

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {1}

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

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

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

ROMV RR | 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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {2}

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

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

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

ROMV RR | 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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {3}

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {3}

ROMV RR | 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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {3}

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {3}

ROMV RR | 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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {3}

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {3}

ROMV RR | 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

ROMV RR | Serialization point. Timestamp: 4

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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {1}

ROMV RR | 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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {1}

ROMV RR | 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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {1}

ROMV RR | 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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {1}

ROMV RR | 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

ROMV RR | 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

ROMV RR | Waiting for locks from transactions: {1}

ROMV RR | 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

ROMV RR | Serialization point. Timestamp: 5

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | Serialization point. Timestamp: 6

ROMV RR |

ROMV RR |

ROMV RR | Data in the end of the run:

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

**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:**

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

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

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

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

ROMV RR | Serialization point. Timestamp: 1

ROMV RR |

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

ROMV RR | Serialization point. Timestamp: 2

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

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

ROMV RR | Waiting for locks from transactions: {2}

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

ROMV RR | Serialization point. Timestamp: 3

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

ROMV RR | Serialization point. Timestamp: 4

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

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

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

ROMV RR | Waiting for locks from transactions: {2}

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

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

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

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

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

ROMV RR | Waiting for locks from transactions: {2}

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

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

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

ROMV RR | Serialization point. Timestamp: 5

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

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

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

ROMV RR | Serialization point. Timestamp: 6

ROMV RR |

ROMV RR |

ROMV RR | Data in the end of the run:

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

**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:**

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

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

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

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

ROMV RR | Serialization point. Timestamp: 1

ROMV RR |

ROMV RR | 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

ROMV RR | Serialization point. Timestamp: 2

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

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

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

ROMV RR | Serialization point. Timestamp: 3

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

ROMV RR | 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

ROMV RR | Serialization point. Timestamp: 4

ROMV RR | 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

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

ROMV RR | 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).

ROMV RR | Serialization point. Timestamp: 5

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

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

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

ROMV RR | 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

ROMV RR | 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

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

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

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

ROMV RR | Waiting for locks from transactions: {3}

ROMV RR | 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

ROMV RR | 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

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

ROMV RR | 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).

ROMV RR | Serialization point. Timestamp: 6

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

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

ROMV RR | 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

ROMV RR | 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

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

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

ROMV RR | 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.

ROMV RR | 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).

delete\_old\_version: y 6

ROMV RR | Serialization point. Timestamp: 7

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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

ROMV RR | 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.

ROMV RR | 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.

ROMV RR | 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.

delete\_old\_version: y 1

delete\_old\_version: x 1

delete\_old\_version: z 1

ROMV RR |

ROMV RR |

ROMV RR | Data in the end of the run:

ROMV RR | {'y': [('9', 7)], 'z': [('9', 7)], 'x': [('19', 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:**

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

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

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

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

ROMV serial | Serialization point. Timestamp: 1

ROMV serial |

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

ROMV serial | Serialization point. Timestamp: 2

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

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

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

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

ROMV serial | 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.

delete\_old\_version: x 1

ROMV serial | Serialization point. Timestamp: 3

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

ROMV serial | Serialization point. Timestamp: 4

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

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

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

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

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

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

ROMV serial | 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.

delete\_old\_version: x 3

ROMV serial | Serialization point. Timestamp: 5

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

ROMV serial | Serialization point. Timestamp: 6

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

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

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

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

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

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

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

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

ROMV serial | 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.

ROMV serial | 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.

delete\_old\_version: z 1

delete\_old\_version: x 5

ROMV serial | Serialization point. Timestamp: 7

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

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

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

ROMV serial | 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.

ROMV serial | 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.

delete\_old\_version: y 1

delete\_old\_version: x 7

ROMV serial | Serialization point. Timestamp: 8

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

ROMV serial | Serialization point. Timestamp: 9

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

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

ROMV serial |

ROMV serial |

ROMV serial | Data in the end of the run:

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

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

\*using a special command "suspend" that allows us to suspend the entrance of the reader to the system.

**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:**

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

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

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

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

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

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

ROMV RR | Serialization point. Timestamp: 1

ROMV RR |

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

ROMV RR | Serialization point. Timestamp: 2

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

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

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

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

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

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

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

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

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

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

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

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

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

ROMV RR | 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).

ROMV RR | Serialization point. Timestamp: 3

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

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

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

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

ROMV RR | 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).

ROMV RR | Serialization point. Timestamp: 4

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

ROMV RR | 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.

ROMV RR | 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.

delete\_old\_version: x 1

delete\_old\_version: u 1

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

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

ROMV RR | Serialization point. Timestamp: 5

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

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

ROMV RR | 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).

ROMV RR | Serialization point. Timestamp: 6

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

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

ROMV RR | Serialization point. Timestamp: 7

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

ROMV RR | 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.

delete\_old\_version: y 1

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

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

ROMV RR |

ROMV RR |

ROMV RR | Data in the end of the run:

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