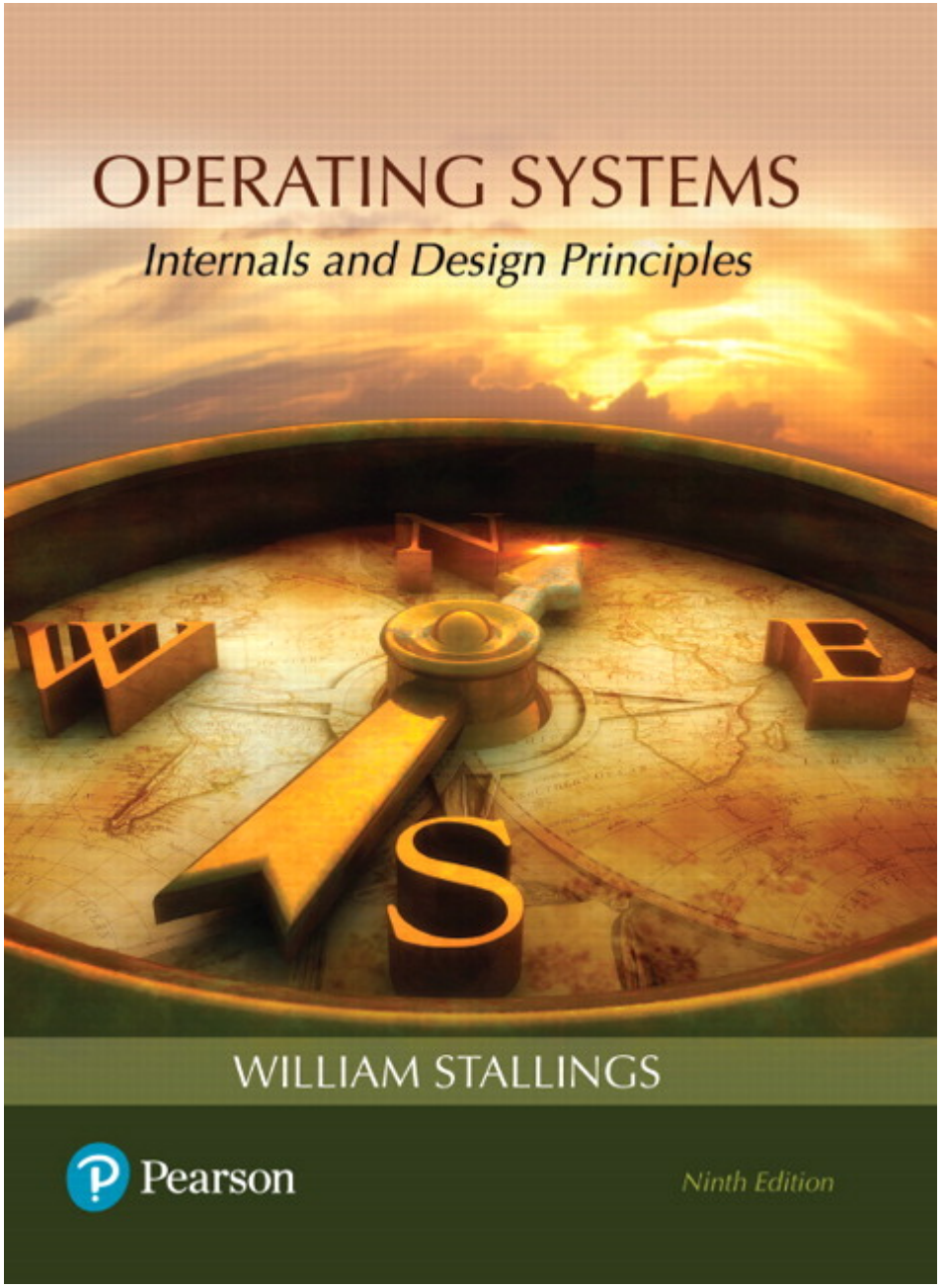


William Stallings

Operating Systems

Internals and Design Principles

Ninth Edition
2017

Readers/Writers Problem, Ver. 2

rdr_wrt_msg_v1.pml (1/7)

```
$ cat -n rdr_wrt_msg_v1.pml | expand
     1  #define NRDRS   5
     2  #define NWRTS   2
     3  #define MAXRDRQ 20
     4  #define MAXWRRQ 20
     5
     6  chan readrequest  = [MAXRDRQ] of { byte, chan }
     7  chan writerequest = [MAXWRRQ] of { byte, chan }
     8  chan finished     = [MAXRDRQ+MAXWRRQ] of { byte }
     9  chan mbox[NRDRS+NWRTS+1] = [MAXRDRQ+MAXWRRQ] of { bool }
    10
    11  byte count = 100
    12  mtype = { reader, writer }
    13  byte nr = 0, nw = 0
    14
...
```

Simplificamos el modelo creando un solo código para los procesos de *Readers* y *Writers*.

```
...
    15  proctype ReaderWriter(byte i; mtype who) {
    16      chan ch
    17      if
    18      :: who == reader -> ch = readrequest
    19      :: else -> ch = writerequest
    20      fi
    21
    22      ch ! i,mbox[i]
    23      atomic {
    24          mbox[i] ? _
    25          printf("%e %d\n",who,i)
    26      }
...
```

rdr_wrt_msg_v1.pml (3/7)

```
...
    27          if
    28          :: who == reader -> nr++
    29          :: else -> nw++
    30          fi
    31          assert(nw < 2)
    32          assert((nw > 0 && nr == 0) || (nw == 0 && nr > 0))
    33          atomic {
    34              if
    35              :: who == reader -> nr--
    36              :: else -> nw--
    37              fi
    38              finished ! i
    39          }
    40  }
    41
...
```

rdr_wrt_msg_v1.pml (4/7)

```
...
    42  proctype Controller() {
    43      byte p
    44
    45  end:
    46      do
    47      ::  count > 0 ->
    48          if
    49          ::  nempty(finished) ->
    50                  atomic {
    51                      finished ? p
    52                      printf("finished %d\n",p)
    53                  }
    54                  count++
    55          ::  empty(finished) && nempty(writerequest) ->
    56                  atomic {
    57                      writerequest ? p
    58                      printf("request from Writer %d\n",p)
    59                  }
    60                  count = count - 100
...
```

rdr_wrt_msg_v1.pml (5/7)

```
...
    61          ::  empty(finished) && empty(writerequest) && nempty(readrequest) ->
    62                  atomic {
    63                      readrequest ? p
    64                      printf("request from Reader %d\n",p)
    65                  }
    66                  count--
    67                  atomic {
    68                      mbox[p] ! true
    69                      printf("OK to Reader %d\n",p)
    70                  }
    71          fi
    72      ::  count == 0 ->
    73              atomic {
    74                  mbox[p] ! true
    75                  printf("OK to Writer %d\n",p)
    76              }
    77              atomic {
    78                  finished ? p
    79                  printf("finished Writer %d\n",p)
    80              }
    81              count = 100
```

rdr_wrt_msg_v1.pml (6/7)

```
...
    82      ::  count < 0 ->
    83              atomic {
    84                  finished ? p
    85                  printf("finished Writer %d\n",p)
    86              }
    87              count++
    88      od
    89  }
    90
...
```

rdr_wrt_msg_v1.pml (7/7)

```
...
 91  init {
 92      byte i
 93
 94      atomic {
 95          for (i : 1 .. NRDRS+NWRTS) {  /* R1,R2,W3,R4,W5,R6,R7 */
 96              if
 97              ::  i == 3 || i == 5 ->
 98                      run ReaderWriter(i,writer)
 99              ::  else ->
100                      run ReaderWriter(i,reader)
101              fi
102          }
103          run Controller()
104      }
105  }
```

Simulation: seed 0 (1/2)

```
$ spin -n0 -B rdr_wrt_msg_v1.pml | expand
                                        request from Writer 5
                                        OK to Writer 5
                            writer 5
                                        finished Writer 5
                                        request from Writer 3
                                        OK to Writer 3
                        writer 3
                                        finished Writer 3
                                        request from Reader 7
                                        OK to Reader 7
                                    reader 7
                                        request from Reader 6
                                        OK to Reader 6
                                reader 6
                                        finished 7
                                        finished 6
                                        request from Reader 4
                                        OK to Reader 4
                            reader 4
...
```

Simulation: seed 0 (2/2)

```
...
                                                request from Reader 2
                                                OK to Reader 2
                reader 2
                                                finished 4
                                                finished 2
                                                request from Reader 1
                                                OK to Reader 1
           reader 1
                                                finished 1
      timeout
```

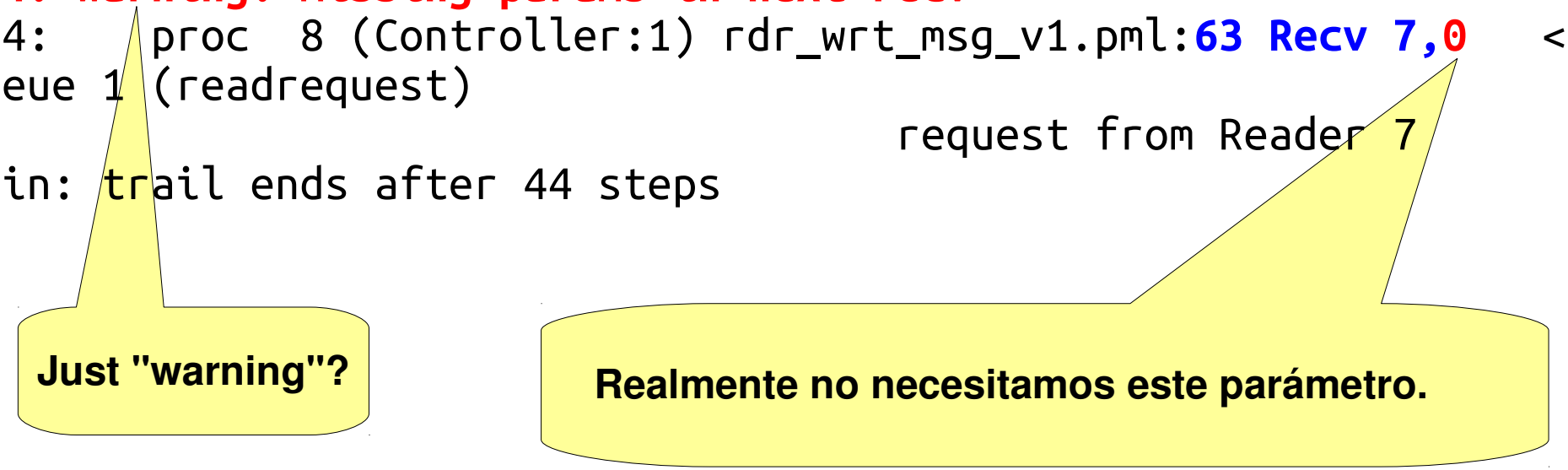
seed 0: W5, W3, (R7+R6), (R4+R2), R1.

Verification: 1 error

```
$ spin -run rdr_wrt_msg_v1.pml | expand
pan:1: missing pars in receive (at depth 43)
pan: wrote rdr_wrt_msg_v1.pml.trail

(Spin Version 6.4.6 -- 2 December 2016)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +

State-vector 620 byte, depth reached 43, errors: 1
...
```

Error trail

```
$ spin -t -s -r -B rdr_wrt_msg_v1.pml | expand

using statement merging
 42:    proc  7 (ReaderWriter:1) rdr_wrt_msg_v1.pml:22 Send 7,9 ->
queue 1 (ch)
 44: warning: missing params in next recv
 44:    proc  8 (Controller:1) rdr_wrt_msg_v1.pml:63 Recv 7,0   <-
queue 1 (readrequest)
                                          request from Reader 7
spin: trail ends after 44 steps
```



Versión 2 del modelo (código unificado para *Readers*/*Writers*)

```
$ cat -n rdr_wrt_msg_v2.pml | expand
     1  #define NRDRS   5
     2  #define NWRTS   2
     3  #define MAXRDRQ 20
     4  #define MAXWRRQ 20
     5
     6  chan readrequest  = [MAXRDRQ] of { byte }
     7  chan writerequest = [MAXWRRQ] of { byte }
     8  chan finished     = [MAXRDRQ+MAXWRRQ] of { byte }
     9  chan mbox[NRDRS+NWRTS+1] = [MAXRDRQ+MAXWRRQ] of { bool }
    10
    11  byte count = 100
    12  mtype = { reader, writer }
    13  byte nr = 0, nw = 0
    14
...
```

```
...
    15  proctype ReaderWriter(byte i; mtype who) {
    16      chan ch
    17      if
    18      :: who == reader -> ch = readrequest
    19      :: else -> ch = writerequest
    20      fi
    21
    22      ch ! i
    23      atomic {
    24          mbox[i] ? _
    25          printf("%e %d\n",who,i)
    26      }
...
```

rdr_wrt_msg_v2.pml (3/7)

```
...
    27      if
    28      :: who == reader -> nr++
    29      :: else -> nw++
    30      fi
    31      assert(nw < 2)
    32      assert((nw > 0 && nr == 0) || (nw == 0 && nr > 0))
    33      atomic {
    34          if
    35          :: who == reader -> nr--
    36          :: else -> nw--
    37          fi
    38          finished ! i
    39      }
    40  }
    41
...
```

rdr_wrt_msg_v2.pml (4/7)

```
...
    42  proctype Controller() {
    43      byte p
    44
    45  end:
    46      do
    47      ::  count > 0 ->
    48          if
    49          ::  nempty(finished) ->
    50                  atomic {
    51                      finished ? p
    52                      printf("finished %d\n",p)
    53                  }
    54                  count++
    55          ::  empty(finished) && nempty(writerequest) ->
    56                  atomic {
    57                      writerequest ? p
    58                      printf("request from Writer %d\n",p)
    59                  }
    60                  count = count - 100
...
```

rdr_wrt_msg_v2.pml (5/7)

```
...
    61          ::  empty(finished) && empty(writerequest) && nempty(readrequest) ->
    62                  atomic {
    63                      readrequest ? p
    64                      printf("request from Reader %d\n",p)
    65                  }
    66                  count--
    67                  atomic {
    68                      mbox[p] ! true
    69                      printf("OK to Reader %d\n",p)
    70                  }
    71          fi
    72      ::  count == 0 ->
    73              atomic {
    74                  mbox[p] ! true
    75                  printf("OK to Writer %d\n",p)
    76              }
    77              atomic {
    78                  finished ? p
    79                  printf("finished Writer %d\n",p)
    80              }
    81              count = 100
```

rdr_wrt_msg_v2.pml (6/7)

```
...
    82      ::  count < 0 ->
    83              atomic {
    84                  finished ? p
    85                  printf("finished Writer %d\n",p)
    86              }
    87              count++
    88      od
    89  }
    90
...
```

rdr_wrt_msg_v2.pml (7/7)

```
...
 91  init {
 92      byte i
 93
 94      atomic {
 95          for (i : 1 .. NRDRS+NWRTS) {  /* R1,R2,W3,R4,W5,R6,R7 */
 96              if
 97              ::  i == 3 || i == 5 ->
 98                      run ReaderWriter(i,writer)
 99              ::  else ->
100                      run ReaderWriter(i,reader)
101              fi
102          }
103          run Controller()
104      }
105  }
```

Verification: 1 error

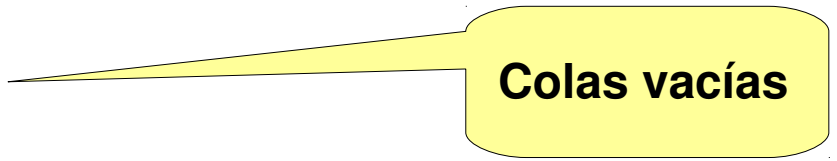
```
$ spin -run rdr_wrt_msg_v2.pml | expand
pan:1: invalid end state (at depth 165)
pan: wrote rdr_wrt_msg_v2.pml.trail

(Spin Version 6.4.6 -- 2 December 2016)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +

State-vector 572 byte, depth reached 166, errors: 1
...
```

Invalid End State Error trail (1/3)

```
$ spin -t rdr_wrt_msg_v2.pml | expand
                                        request from Reader 7
                                          OK to Reader 7
                                      reader 7
                                          finished 7
                                          request from Reader 6
                                          OK to Reader 6
                                    reader 6
                                          finished 6
                                          request from Writer 5
                                          OK to Writer 5
                              writer 5
                                          finished Writer 5
                                          request from Reader 4
                                          OK to Reader 4
                          reader 4
                                          finished 4
                                          request from Writer 3
                                          OK to Writer 3
                     Writer 3
...
```

Invalid End State Error trail (2/3)

```
...
                                                finished Writer 3
                                                request from Reader 2
                                                OK to Reader 2
                  reader 2
                                                finished 2
                                                request from Reader 1
                                                OK to Reader 1
             reader 1
                                                finished 1
spin: trail ends after 166 steps
#processes: 9
                    queue 1 (readrequest):
                    queue 3 (writerequest):
                    queue 2 (finished):
                    queue 4 (mbox[0]):
                    queue 5 (mbox[1]):
                    queue 6 (mbox[2]):
                    queue 7 (mbox[3]):
                    queue 8 (mbox[4]):
                    queue 9 (mbox[5]):
...
```



Invalid End State Error trail (3/3)

```
...
                queue 10 (mbox[6]):
                queue 11 (mbox[7]):
                count = 100
                nr = 0
                nw = 0
166:    proc  8 (Controller:1) rdr_wrt_msg_v2.pml:48 (state 20)
166:    proc  7 (ReaderWriter:1) rdr_wrt_msg_v2.pml:40 (state 27) <valid end state>
166:    proc  6 (ReaderWriter:1) rdr_wrt_msg_v2.pml:40 (state 27) <valid end state>
166:    proc  5 (ReaderWriter:1) rdr_wrt_msg_v2.pml:40 (state 27) <valid end state>
166:    proc  4 (ReaderWriter:1) rdr_wrt_msg_v2.pml:40 (state 27) <valid end state>
166:    proc  3 (ReaderWriter:1) rdr_wrt_msg_v2.pml:40 (state 27) <valid end state>
166:    proc  2 (ReaderWriter:1) rdr_wrt_msg_v2.pml:40 (state 27) <valid end state>
166:    proc  1 (ReaderWriter:1) rdr_wrt_msg_v2.pml:40 (state 27) <valid end state>
166:    proc  0 (:init::1) rdr_wrt_msg_v2.pml:105 (state 17) <valid end state>
9 processes created
```

Invalid end state

End label is misplaced:
45 → 48

rdr_wrt_msg_v3.pml (4/7 only)

```
...
    42  proctype Controller() {
    43      byte p
    44
    45      do
    46      ::  count > 0 ->
    47  end:    if
    48          ::  nempty(finished) ->
    49                  atomic {
    50                      finished ? p
    51                      printf("finished %d\n",p)
    52                  }
    53                  count++
    54          ::  empty(finished) && nempty(writerequest) ->
    55                  atomic {
    56                      writerequest ? p
    57                      printf("request from Writer %d\n",p)
    58                  }
    59                  count = count - 100
...
```

Versión 3 del modelo (end label correct place)

Verification: 1 error

```
$ spin -run rdr_wrt_msg_v3.pml | expand
pan:1: invalid end state (at depth 138)
pan: wrote rdr_wrt_msg_v3.pml.trail

(Spin Version 6.4.6 -- 2 December 2016)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +

State-vector 572 byte, depth reached 166, errors: 1
...
```

Invalid End State Error trail (1/3)

```
$ spin -t rdr_wrt_msg_v3.pml | expand
                                        request from Reader 7
                                        OK to Reader 7
                                    reader 7
                                        finished 7
                                        request from Reader 6
                                        OK to Reader 6
                                reader 6
                                        finished 6
                                        request from Writer 5
                                        OK to Writer 5
                            writer 5
                                        finished Writer 5
                                        request from Reader 4
                                        OK to Reader 4
                        reader 4
                                        finished 4
                                        request from Reader 2
                                        OK to Reader 2
                                        request from Writer 3
...
```

Invalid End State Error trail (2/3)

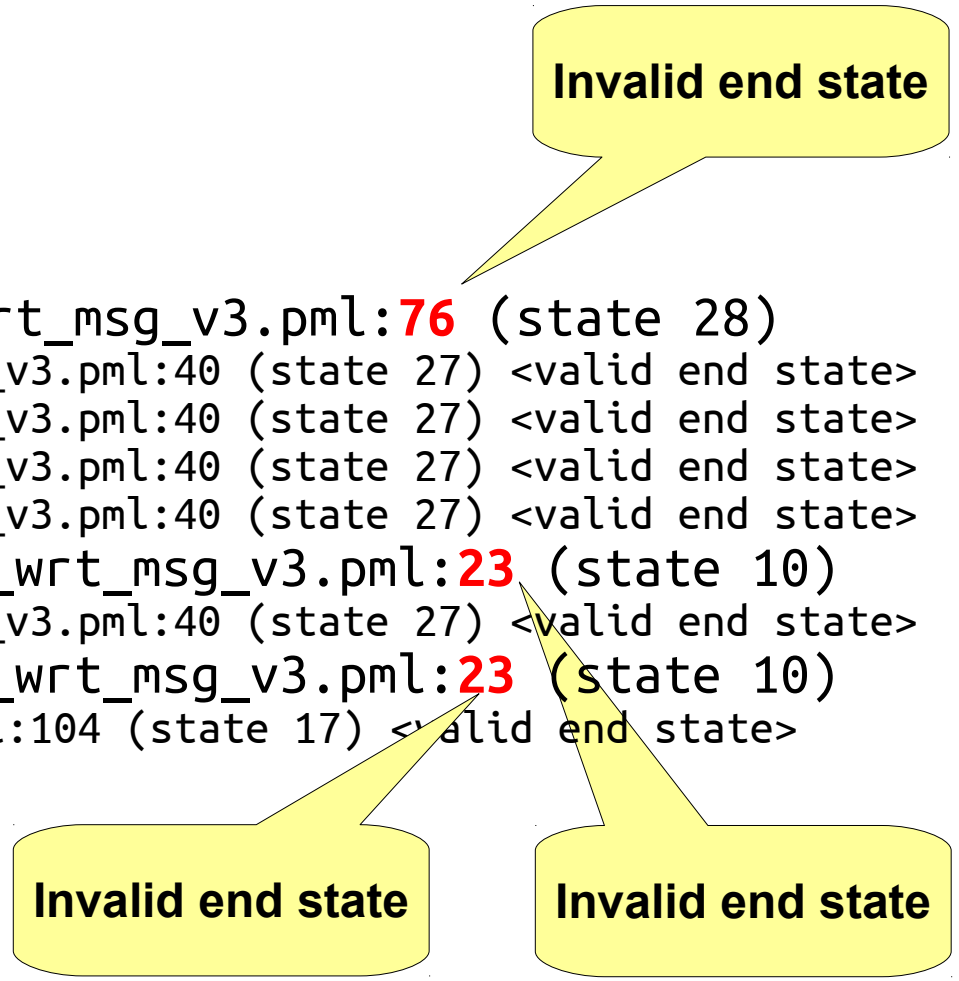
```
...
spin: rdr_wrt_msg_v3.pml:59, Error: value (-1->255 (8)) truncated in assignment
                reader 2
                                        finished 2
spin: rdr_wrt_msg_v3.pml:53, Error: value (256->0 (8)) truncated in assignment
                                        OK to Writer 2
spin: trail ends after 139 steps
#processes: 9
                queue 1 (readrequest): [1]
                queue 3 (writerequest):
                queue 2 (finished):
                queue 4 (mbox[0]):
                queue 5 (mbox[1]):
                queue 6 (mbox[2]): [1]
                queue 7 (mbox[3]):
                queue 8 (mbox[4]):
                queue 9 (mbox[5]):
                queue 10 (mbox[6]):
                queue 11 (mbox[7]):
                count = 0
                nr = 0
                nw = 0
...
```

Esto es lo más preocupante

Tambien colas no procesadas

Invalid End State Error trail (3/3)

```
...
139:    proc  8 (Controller:1) rdr_wrt_msg_v3.pml:76 (state 28)
139:    proc  7 (ReaderWriter:1) rdr_wrt_msg_v3.pml:40 (state 27) <valid end state>
139:    proc  6 (ReaderWriter:1) rdr_wrt_msg_v3.pml:40 (state 27) <valid end state>
139:    proc  5 (ReaderWriter:1) rdr_wrt_msg_v3.pml:40 (state 27) <valid end state>
139:    proc  4 (ReaderWriter:1) rdr_wrt_msg_v3.pml:40 (state 27) <valid end state>
139:    proc  3 (ReaderWriter:1) rdr_wrt_msg_v3.pml:23 (state 10)
139:    proc  2 (ReaderWriter:1) rdr_wrt_msg_v3.pml:40 (state 27) <valid end state>
139:    proc  1 (ReaderWriter:1) rdr_wrt_msg_v3.pml:23 (state 10)
139:    proc  0 (:init::1) rdr_wrt_msg_v3.pml:104 (state 17) <valid end state>
9 processes created
```



Observaciones

Se suponía que el algoritmo garantiza el procesamiento prioritario de las solicitudes de los *Writers*. Para este propósito sirve la variable **`count`**. Pero parece que su manejo no es correcto.

No nos queda otra cosa que encontrar el error y desarrollar la siguiente versión del modelo: **`rdr_wrt_msg_v4.pml`**.