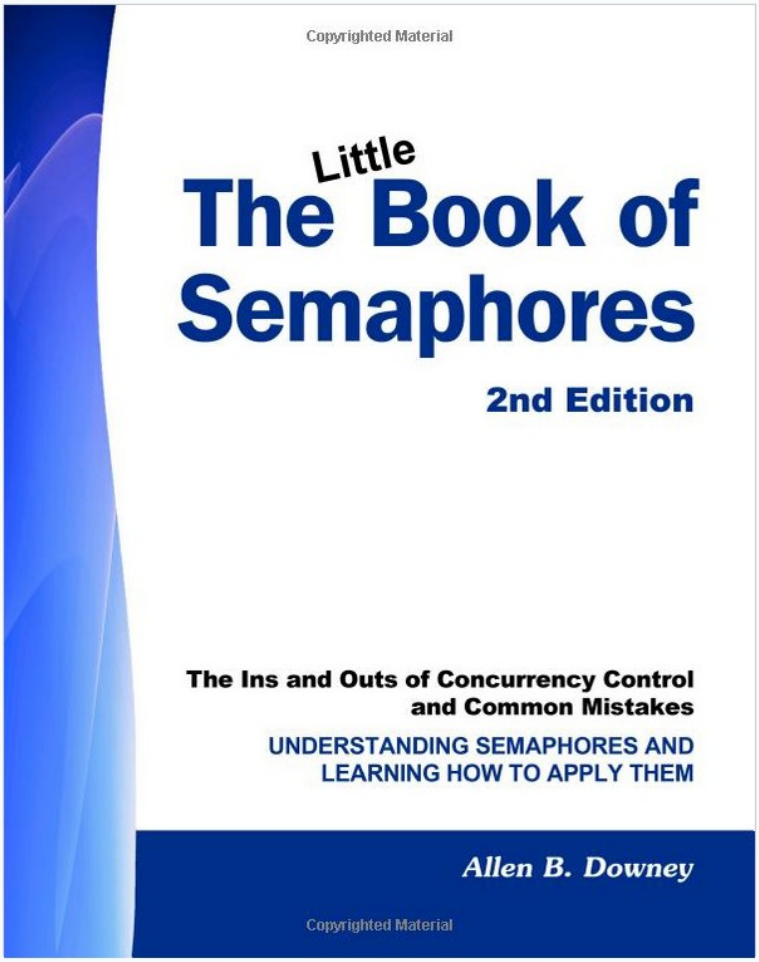

Allen B. Downey

The Little Book of Semaphores

Version 2.2.1

http://www.greenteapress.com/semaphores/LittleBookOfSemaphores.pdf

3.7.1 Reusable barrier non-solution (3.7.1a.rebarrier_nonsol.pml)

```
$ cat -n 3.7.1a.rebarrier_nonsol.pml | expand

     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.7 Reusable barrier
     7      3.7.1 Reusable barrier non-solution
     8
     9      vk, 2017
    10  */
    11
    12  #define THREADS 3    /* value for threads number */
    13  #define N       3    /* value for barrier limit */
    14
    15  #define wait(sem)   atomic { sem > 0; sem-- }
    16  #define signal(sem) sem++
    17
    18  byte count=0, mutex=1, turnstile=0    /* turnstile is locked */
    19
...
```


3.7.1 Reusable barrier non-solution (3.7.1a.rebarrier_nonsol.pml)

```
...
    20  proctype Th(byte i) {
    21      byte temp
    22
    23  rendezvous:
    24      do
    25      ::  wait(mutex)
    26              temp=count
    27              count=temp+1
    28          signal(mutex)
    29          if
    30          :: count == N ->
    31              signal(turnstile)
    32          :: else
    33          fi
    34          wait(turnstile)
    35          printf("Th(%d): count = %d\n",i,count)
    36          signal(turnstile)
...
```

3.7.1 Reusable barrier non-solution (3.7.1a.rebarrier_nonsol.pml)

```
...
    37  critical:
    38          wait(mutex)
    39              temp=count
    40              count=temp-1
    41          signal(mutex)
    42          if
    43          :: count == 0 ->
    44              wait(turnstile)
    45          :: else
    46          fi
    47          break    /* one only iteration */
    48      od
    49  }
    50
...
```


3.7.1 Reusable barrier non-solution (3.7.1a.rebarrier_nonsol.pml)

```
$ spin 3.7.1a.rebarrier_nonsol.pml | expand
                    Th(3): count = 3
                Th(2): count = 3
            Th(1): count = 3
        turnstile = 0
4 processes created


$ spin 3.7.1a.rebarrier_nonsol.pml | expand
            Th(1): count = 3
                    Th(3): count = 3
                Th(2): count = 2
        turnstile = 0
4 processes created


$ spin 3.7.1a.rebarrier_nonsol.pml | expand
                    Th(3): count = 3
                Th(2): count = 3
            Th(1): count = 1
        turnstile = 0
4 processes created
```

3.7.1 Reusable barrier non-solution (3.7.1b.rebarrier_nonsol.pml)

```
...
    51  init {
    52      byte i
    53
    54      atomic {
    55          for (i: 1 .. THREADS) {
    56              run Th(i)
    57          }
    58      }
    59      _nr_pr == 1 ->
    60          assert(turnstile == 0)
    61          printf("turnstile = %d\n",turnstile)
    62  }
```


3.7.1 Reusable barrier non-solution (3.7.1b.rebarrier_nonsol.pml)

```
$ spin 3.7.1b.rebarrier_nonsol.pml | expand
                    Th(3): count = 3
                Th(2): count = 2
            Th(1): count = 2
        turnstile = 0
4 processes created


$ spin 3.7.1b.rebarrier_nonsol.pml | expand
                Th(2): count = 3
                    Th(3): count = 3
            Th(1): count = 3
spin: 3.7.1b.rebarrier_nonsol.pml:60, Error: assertion violated
spin: text of failed assertion: assert((turnstile==0))
#processes: 1
                    count = 0
                    mutex = 1
                    turnstile = 1
 76:    proc  0 (:init::1) 3.7.1b.rebarrier_nonsol.pml:60 (state 12)
4 processes created
```

3.7.1 Reusable barrier non-solution (3.7.1c.rebarrier_nonsol.pml)

```
...
    51  init {
    52      byte i
    53
    54      atomic {
    55          for (i: 1 .. THREADS) {
    56              run Th(i)
    57          }
    58      }
    59      _nr_pr == 1 ->
    60          assert(turnstile < 3)
    61          printf("turnstile = %d\n",turnstile)
    62  }
```


3.7.1 Reusable barrier non-solution (3.7.1c.rebarrier_nonsol.pml)

```
$ spin 3.7.1c.rebarrier_nonsol.pml | expand
pan:1: invalid end state (at depth 55)
pan: wrote 3.7.1c.rebarrier_nonsol.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 48 byte, depth reached 61, errors: 1
       66 states, stored
        5 states, matched
       71 transitions (= stored+matched)
       12 atomic steps
hash conflicts:         0 (resolved)
...
```

3.7.1 Reusable barrier non-solution (3.7.1c.rebarrier_nonsol.pml)

```
$ spin -t -p -g -l 3.7.1c.rebarrier_nonsol.pml | expand
...
Starting Th with pid 1 ...
Starting Th with pid 2 ...
Starting Th with pid 3 ...
 28:    proc  1 (Th:1) 3.7.1c.rebarrier_nonsol.pml:30 (state 7) [((count==3))]
 29:    proc  1 (Th:1) 3.7.1c.rebarrier_nonsol.pml:31 (state 8) [turnstile = (turnstile+1)]
                turnstile = 1
...
 38: proc 3 terminates
...
              Th(2): count = 2
...
          Th(1): count = 1
...
 54: proc 2 terminates

 55:    proc  1 (Th:1) 3.7.1c.rebarrier_nonsol.pml:41 (state 22)        [mutex = (mutex+1)]
                mutex = 1
 56:    proc  1 (Th:1) 3.7.1c.rebarrier_nonsol.pml:43 (state 23)        [((count==0))]
spin: trail ends after 56 steps
#processes: 2
                count = 0
                mutex = 1
                turnstile = 0
 56:    proc  1 (Th:1) 3.7.1c.rebarrier_nonsol.pml:44 (state 26)
 56:    proc  0 (:init::1) 3.7.1c.rebarrier_nonsol.pml:59 (state 11)
4 processes created
...
```

1 process blocked!
2 processes blocked?
All 3 processes blocked?

3.7.1 Reusable barrier non-solution (3.7.1c.rebarrier_nonsol.pml)

```
$ spin -run -E 3.7.1c.rebarrier_nonsol.pml | expand

(Spin Version 6.4.6 -- 2 December 2016)
        + Partial Order Reduction

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

State-vector 48 byte, depth reached 63, errors: 0
    16071 states, stored
    20784 states, matched
    36855 transitions (= stored+matched)
       12 atomic steps
hash conflicts:         7 (resolved)
...
```

3.7.3 Reusable barrier non-solution (3.7.3a.rebarrier_nonsol.pml)

```
$ cat -n 3.7.3a.rebarrier_nonsol.pml | expand

     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.7 Reusable barrier
     7      3.7.3 Reusable barrier non-solution #2
     8
     9      vk, 2017
    10  */
    11
    12  #define THREADS 3    /* value for threads number */
    13  #define N       3    /* value for barrier limit */
    14
    15  #define wait(sem)   atomic { sem > 0; sem-- }
    16  #define signal(sem) sem++
    17
    18  byte count=0, mutex=1, turnstile=0    /* turnstile is locked */
    19
...
```


3.7.3 Reusable barrier non-solution (3.7.3a.rebarrier_nonsol.pml)

```
...
    20  proctype Th(byte i) {
    21      byte temp
    22
    23  rendezvous:
    24      do
    25      ::  wait(mutex)
    26              temp=count
    27              count=temp+1
    28          if
    29          :: count == N ->    /* may be true for one thread only */
    30              signal(turnstile)
    31          :: else
    32          fi
    33          signal(mutex)
    34
    35          wait(turnstile)
    36          signal(turnstile)
    37          printf("Th(%d): count = %d, turnstile = %d\n",
                       i,count,turnstile)
...
```

3.7.3 Reusable barrier non-solution (3.7.3a.rebarrier_nonsol.pml)

```
...
    38  critical:
    39          wait(mutex)
    40              temp=count
    41              count=temp-1
    42          if
    43          :: count == 0 ->    /* may be true for one threads only */
    44              wait(turnstile)    /* leave turnstile locked */
    45          :: else
    46          fi
    47          signal(mutex)
    48          break    /* one only iteration */
    49      od
    50  }
    51
...
```


3.7.3 Reusable barrier non-solution (3.7.3a.rebarrier_nonsol.pml)

```
...
    52  init {
    53      byte i
    54
    55      atomic {
    56          for (i: 1 .. THREADS) {
    57              run Th(i)
    58          }
    59      }
    60      _nr_pr == 1 ->
    61          assert(turnstile == 0)
    62          printf("turnstile = %d\n",turnstile)
    63  }
```

3.7.3 Reusable barrier non-solution (3.7.3a.rebarrier_nonsol.pml)

```
$ spin -run 3.7.3a.rebarrier_nonsol.pml | expand

(Spin Version 6.4.6 -- 2 December 2016)
        + Partial Order Reduction

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

State-vector 48 byte, depth reached 61, errors: 0
...
```


3.7.3 Reusable barrier non-solution (3.7.3a.rebarrier_nonsol.pml)

```
$ spin 3.7.3a.rebarrier_nonsol.pml | expand
              Th(2): count = 3, turnstile = 1
          Th(1): count = 3, turnstile = 1
                  Th(3): count = 2, turnstile = 1
      turnstile = 0
4 processes created

$ spin 3.7.3a.rebarrier_nonsol.pml | expand
                  Th(3): count = 3, turnstile = 1
          Th(1): count = 3, turnstile = 0
              Th(2): count = 3, turnstile = 1
      turnstile = 0
4 processes created

$ spin 3.7.3a.rebarrier_nonsol.pml | expand
                  Th(3): count = 3, turnstile = 0
              Th(2): count = 3, turnstile = 0
          Th(1): count = 1, turnstile = 0
      turnstile = 0
4 processes created
```

3.7.3 Reusable barrier non-solution (3.7.3b.rebarrier_nonsol.pml)

```
$ cat -n 3.7.3b.rebarrier_nonsol.pml | expand

     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.7 Reusable barrier
     7      3.7.3 Reusable barrier non-solution #2
     8
     9      vk, 2017
    10  */
    11
    12  #define THREADS 3    /* value for threads number */
    13  #define N       3    /* value for barrier limit */
    14
    15  #define wait(sem)   atomic { sem > 0; sem-- }
    16  #define signal(sem) sem++
    17
    18  byte count=0, mutex=1, turnstile=0    /* turnstile is locked */
    19  byte loop[THREADS+1]=1
    20  bool sameloop=true
...
```


3.7.3 Reusable barrier non-solution (3.7.3b.rebarrier_nonsol.pml)

```
...
    22  proctype Th(byte i) {
    23      byte temp, j
    24
    25  rendezvous:
    26      do
    27      ::  wait(mutex)
    28              temp=count
    29              count=temp+1
    30          if
    31          :: count == N ->    /* may be true for one thread only */
    32              signal(turnstile)
    33          :: else
    34          fi
    35          signal(mutex)
    36
    37          wait(turnstile)
    38          signal(turnstile)
    39          printf("Th(%d): loop %d\n",i,loop[i])
    40
...
```

3.7.3 Reusable barrier non-solution (3.7.3b.rebarrier_nonsol.pml)

```
...
    41  critical:
    42          atomic {
    43              for (j: 1 .. N-1) {
    44                  sameloop = sameloop && (loop[j] == loop[j+1])
    45              }
    46              assert(sameloop)
    47          }
    48
    49          wait(mutex)
    50              temp=count
    51              count=temp-1
    52          if
    53          :: count == 0 ->    /* may be true for one threads only */
    54              wait(turnstile)    /* leave turnstile locked */
    55          :: else
    56          fi
    57          signal(mutex)
    58
...
```


3.7.3 Reusable barrier non-solution (3.7.3b.rebarrier_nonsol.pml)

```
...
    59          if
    60          :: loop[i] == 2 ->
    61              break
    62          :: else ->
    63              loop[i]++
    64          fi
    65      od
    66  }
    67
    68  init {
    69      byte i
    70
    71      atomic {
    72          for (i: 1 .. THREADS) {
    73              run Th(i)
    74          }
    75      }
    76      _nr_pr == 1 ->
    77          assert(turnstile == 0)
    78          printf("turnstile = %d\n",turnstile)
    79  }
```

3.7.3 Reusable barrier non-solution (3.7.3b.rebarrier_nonsol.pml)

```
$ spin 3.7.3b.rebarrier_nonsol.pml | expand
          Th(1): loop 1
              Th(2): loop 1
                  Th(3): loop 1
          Th(1): loop 2
spin: 3.7.3b.rebarrier_nonsol.pml:46, Error: assertion violated
spin: text of failed assertion: assert(sameloop)
#processes: 4
                count = 1
                mutex = 0
                turnstile = 1
                loop[0] = 1
                loop[1] = 2
                loop[2] = 2
                loop[3] = 1
                sameloop = 0
128:    proc  3 (Th:1) 3.7.3b.rebarrier_nonsol.pml:52 (state 38)
128:    proc  2 (Th:1) 3.7.3b.rebarrier_nonsol.pml:26 (state 47)
128:    proc  1 (Th:1) 3.7.3b.rebarrier_nonsol.pml:46 (state 26)
128:    proc  0 (:init::1) 3.7.3b.rebarrier_nonsol.pml:76 (state 11)
4 processes created
```


3.7.5 Reusable barrier solution (3.7.5.rebarrier.pml)

```
$ cat -n 3.7.5.rebarrier.pml | expand
     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.7 Reusable barrier
     7      3.7.5 Reusable barrier solution
     8
     9      vk, 2017
    10  */
    11
    12  #define THREADS 3    /* value for threads number */
    13  #define N       3    /* value for barrier limit */
    14
    15  #define wait(sem)   atomic { sem > 0; sem-- }
    16  #define signal(sem) sem++
    17
    18  byte count=0, mutex=1, turnstile=0, turnstile2=1
    19  byte loop[THREADS+1]=1
    20  bool sameloop=true
    21
...
```

3.7.5 Reusable barrier solution (3.7.5.rebarrier.pml)

```
...
    22  proctype Th(byte i) {
    23      byte temp, j
    24
    25  rendezvous:
    26      do
    27      ::  wait(mutex)
    28              temp=count
    29              count=temp+1
    30          if
    31          :: count == N ->
    32              wait(turnstile2)    /* lock the second */
    33              signal(turnstile)   /* unlock the first */
    34          :: else
    35          fi
    36          signal(mutex)
    37
    38          wait(turnstile)         /* first turnstile */
    39          signal(turnstile)
    40          printf("Th(%d): loop %d\n",i,loop[i])
    41
...
```


3.7.5 Reusable barrier solution (3.7.5.rebarrier.pml)

```
...
    42  critical:
    43          atomic {
    44              for (j: 1 .. N-1) {
    45                  sameloop = sameloop && (loop[j] == loop[j+1])
    46              }
    47              assert(sameloop)
    48          }
    49
    50          wait(mutex)
    51              temp=count
    52              count=temp-1
    53          if
    54          :: count == 0 ->
    55              wait(turnstile)     /* lock the first */
    56              signal(turnstile2)  /* unlock the second */
    57          :: else
    58          fi
    59          signal(mutex)
    60
    61          wait(turnstile2)        /* second turnstile */
    62          signal(turnstile2)
...
```

3.7.5 Reusable barrier solution (3.7.5.rebarrier.pml)

```
...
    63          if
    64          :: loop[i] == 3 ->
    65              break
    66          :: else ->
    67              loop[i]++
    68          fi
    69      od
    70  }
    71
    72  init {
    73      byte i
    74
    75      atomic {
    76          for (i: 1 .. THREADS) {
    77              run Th(i)
    78          }
    79      }
    80      _nr_pr == 1 ->
    81          assert(turnstile == 0)
    82          assert(turnstile2 == 1)
    83  }
```


3.7.5 Reusable barrier solution (3.7.5.rebarrier.pml)

```
$ spin 3.7.5.rebarrier.pml | expand
          Th(1): loop 1
                  Th(3): loop 1
              Th(2): loop 1
              Th(2): loop 2
                  Th(3): loop 2
          Th(1): loop 2
              Th(2): loop 3
          Th(1): loop 3
                  Th(3): loop 3
4 processes created

$ spin -run 3.7.5.rebarrier.pml | expand
(Spin Version 6.4.6 -- 2 December 2016)
        + Partial Order Reduction

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

State-vector 48 byte, depth reached 236, errors: 0
...
```

3.7.6 Preloaded turnstile (3.7.6.rebarrier_preloaded.pml)

```
$ cat -n 3.7.6.rebarrier_preloaded.pml | expand
     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.7 Reusable barrier
     7      3.7.6 Preloaded turnstile
     8
     9      vk, 2017
    10  */
    11
    12  #define THREADS 3    /* value for threads number */
    13  #define N       3    /* value for barrier limit */
    14
    15  #define wait(sem)   atomic { sem > 0; sem-- }
    16  #define signal(sem) sem++
    17  #define signalN(sem,NN) for (j: 1 .. NN) { sem++ } /* no atomic */
    18
    19  byte count=0, mutex=1, turnstile=0, turnstile2=0
    20  byte loop[THREADS+1]=1
    21  bool sameloop=true
...
```


3.7.6 Preloaded turnstile (3.7.6.rebarrier_preloaded.pml)

```
...
    22
    23  proctype Th(byte i) {
    24      byte temp, j
    25
    26  rendezvous:
    27      do
    28      ::  wait(mutex)
    29              temp=count
    30              count=temp+1
    31          if
    32          :: count == N ->
    33              signalN(turnstile,N)   /* unlock the first */
    34          :: else
    35          fi
    36          signal(mutex)
    37
    38          wait(turnstile)         /* first turnstile */
    39          printf("Th(%d): loop %d\n",i,loop[i])
    40
...
```

3.7.6 Preloaded turnstile (3.7.6.rebarrier_preloaded.pml)

```
...
    41  critical:
    42          atomic {
    43              for (j: 1 .. N-1) {
    44                  sameloop = sameloop && (loop[j] == loop[j+1])
    45              }
    46              assert(sameloop)
    47          }
    48
    49          wait(mutex)
    50              temp=count
    51              count=temp-1
    52          if
    53          :: count == 0 ->
    54              signalN(turnstile2,N) /* unlock the second */
    55          :: else
    56          fi
    57          signal(mutex)
    58
    59          wait(turnstile2)        /* second turnstile */
...
```


3.7.6 Preloaded turnstile (3.7.6.rebarrier_preloaded.pml)

```
...
    60          if
    61          :: loop[i] == 3 ->
    62              break
    63          :: else ->
    64              loop[i]++
    65          fi
    66      od
    67  }
    68
    69  init {
    70      byte i
    71
    72      atomic {
    73          for (i: 1 .. THREADS) {
    74              run Th(i)
    75          }
    76      }
    77      _nr_pr == 1 ->
    78          assert(turnstile == 0)
    79          assert(turnstile2 == 0)
    80  }
```

3.7.6 Preloaded turnstile (3.7.6.rebarrier_preloaded.pml)

```
$ spin -run 3.7.6.rebarrier_preloaded.pml | expand
(Spin Version 6.4.6 -- 2 December 2016)
        + Partial Order Reduction

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

State-vector 48 byte, depth reached 272, errors: 0

$ spin 3.7.6.rebarrier_preloaded.pml | expand
             Th(2): loop 1
          Th(1): loop 1
                 Th(3): loop 1
          Th(1): loop 2
                 Th(3): loop 2
             Th(2): loop 2
                 Th(3): loop 3
          Th(1): loop 3
             Th(2): loop 3
4 processes created
```


3.7.7 Barrier objects (3.7.7.barrier_object.pml)

```
$ cat -n 3.7.7.barrier_object.pml | expand
     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.7 Reusable barrier
     7      3.7.7 Barrier objects
     8
     9      vk, 2017
    10  */
    11
    12  #include "Semaphore.h"
    13  #include "Barrier.h"
    14
    15  #define THREADS 3    /* value for threads number */
    16  #define N       3    /* value for barrier limit */
    17
    18  Semaphore mutex=1
    19  Barrier   barrier
    20
    21  byte loop[THREADS+1]=1
    22  unsigned group : 31 = 0
    23
```

3.7.7 Barrier objects (3.7.7.barrier_object.pml)

```
...
    24  proctype Th(byte i) {
    25      do
    26      ::
    27          printf("Th(%d): loop %d\n",i,loop[i])
    28  rendezvous:
    29          bar_wait(barrier)
    30  critical:
    31          group=group*10+loop[i]
    32          assert(group==1 || group==11 || group==111 ||
    33                 group==1112 || group==11122 || group==111222 ||
    34                 group==1112223 || group==11122233 || group==111222333)
    35          printf("Th(%d): loop %d passed with %d\n",i,loop[i],group)
    36
    37          if
    38          :: loop[i] == 3 ->
    39              break
    40          :: else ->
    41              loop[i]++
    42          fi
    43      od
    44  }
    45
...
```


3.7.7 Barrier objects (3.7.7.barrier_object.pml)

```
...
    46  init {
    47      byte i
    48
    49      bar_init(barrier,N)
    50
    51      atomic {
    52          for (i: 1 .. THREADS) {
    53              run Th(i)
    54          }
    55      }
    56      _nr_pr == 1 ->
    57          assert(barrier._turnstile == 0)
    58          assert(barrier._turnstile2 == 0)
    59  }
```

3.7.7 Barrier objects (3.7.7.barrier_object.pml)

```
$ cat -n Semaphore.h | expand
     1
     2  #define Semaphore   byte
     3
     4  #define wait(sem)       atomic { sem > 0; sem-- }
     5  #define signal(sem)     sem++
     6  #define signalN(sem,NN) for (_i: 1 .. NN) { sem++ } /* no atomic */
     7
     8  byte _i=0
     9
```


3.7.7 Barrier objects (3.7.7.barrier_object.pml)

```
$ cat -n Barrier.h | expand
     1
     2  typedef Barrier {
     3      byte      _n
     4      byte      _count
     5      Semaphore _mutex
     6      Semaphore _turnstile
     7      Semaphore _turnstile2
     8  }
     9
    10  inline bar_init(bar,n) {
    11      bar._n          = n
    12      bar._count      = 0
    13      bar._mutex      = 1
    14      bar._turnstile  = 0
    15      bar._turnstile2 = 0
    16  }
    17
...
```

3.7.7 Barrier objects (3.7.7.barrier_object.pml)

```
...
    18  inline bar_phase1(bar) {
    19      wait(bar._mutex)
    20          bar._count++    /* atomic here */
    21          if
    22          :: bar._count == bar._n ->
    23              signalN(bar._turnstile,bar._n)
    24          :: else
    25          fi
    26      signal(bar._mutex)
    27      wait(bar._turnstile)
    28  }
    29
...
```


3.7.7 Barrier objects (3.7.7.barrier_object.pml)

```
...
    30  inline bar_phase2(bar) {
    31      wait(bar._mutex)
    32          bar._count--        /* atomic here */
    33          if
    34          :: bar._count == 0 ->
    35              signalN(bar._turnstile2,bar._n)
    36          :: else
    37          fi
    38      signal(bar._mutex)
    39      wait(bar._turnstile2)
    40  }
    41
    42  inline bar_wait(bar) {
    43      bar_phase1(bar)
    44      bar_phase2(bar)
    45  }
    46
```

3.7.7 Barrier objects (3.7.7.barrier_object.pml)

```
$ spin -T 3.7.7.barrier_object.pml | expand

Th(1): loop 1
Th(2): loop 1
Th(3): loop 1
Th(2): loop 1 passed with 1
Th(1): loop 1 passed with 11
Th(2): loop 2
Th(3): loop 1 passed with 111
Th(1): loop 2
Th(3): loop 2
Th(1): loop 2 passed with 11122
Th(2): loop 2 passed with 11122
Th(2): loop 3
Th(3): loop 2 passed with 111222
Th(1): loop 3
Th(3): loop 3
Th(2): loop 3 passed with 1112223
Th(1): loop 3 passed with 11122233
Th(3): loop 3 passed with 111222333
4 processes created
```


3.7.7 Barrier objects (3.7.7.barrier_object.pml)

```
$ spin -run 3.7.7.barrier_object.pml | expand

(Spin Version 6.4.6 -- 2 December 2016)
        + Partial Order Reduction

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

State-vector 52 byte, depth reached 232, errors: 0
...
```

3.8.4 Exclusive queue (3.8.4a.exclusive_queue.pml)

```
$ cat -n 3.8.4a.exclusive_queue.pml | expand

     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.8 Queue
     7      3.8.4 Exclusive queue solution
     8
     9      vk, 2017
    10  */
    11
    12  #include "Semaphore.h"
    13
    14  #define N 6
    15
    16  Semaphore mutex=1, leaderQueue=0, followerQueue=0, rendezvous=0
    17  byte      leaders=0, followers=0
    18
...
```


3.8.4 Exclusive queue (3.8.4a.exclusive_queue.pml)

```
...
    19  proctype Leader(byte i) {
    20      wait(mutex)
    21      if
    22      :: followers > 0 ->
    23          followers--
    24          signal(followerQueue)
    25      :: else ->
    26          leaders++
    27          signal(mutex)
    28          wait(leaderQueue)
    29      fi
    30
    31  dance:
    32      printf("leader %d: to dance\n",i)
    33      wait(rendezvous)
    34      printf("leader %d: dancing\n",i)
    35      signal(mutex)
    36  }
    37
...
```

3.8.4 Exclusive queue (3.8.4a.exclusive_queue.pml)

```
...
    38  proctype Follower(byte i) {
    39      wait(mutex)
    40      if
    41      :: leaders > 0 ->
    42          leaders--
    43          signal(leaderQueue)
    44      :: else ->
    45          followers++
    46          signal(mutex)
    47          wait(followerQueue)
    48      fi
    49
    50  dance:
    51      atomic {
    52          signal(rendezvous)
    53          printf("follower %d: dancing\n",i)
    54      }
    55  }
    56
...
```


3.8.4 Exclusive queue (3.8.4a.exclusive_queue.pml)

```
...
    57  init {
    58      byte i
    59
    60      atomic {
    61          for (i: 1 .. N) {
    62              if
    63              :: i % 2 -> run Leader(i)
    64              :: else  -> run Follower(i)
    65              fi
    66          }
    67      }
    68  }
```

3.8.4 Exclusive queue (3.8.4a.exclusive_queue.pml)

```
$ spin -T 3.8.4a.exclusive_queue.pml | expand
follower 2: dancing
leader 1: to dance
leader 1: dancing
follower 6: dancing
leader 3: to dance
leader 3: dancing
follower 4: dancing
leader 5: to dance
leader 5: dancing
7 processes created

$ spin -T 3.8.4a.exclusive_queue.pml | expand
follower 2: dancing
leader 5: to dance
leader 5: dancing
leader 3: to dance
follower 6: dancing
leader 3: dancing
leader 1: to dance
follower 4: dancing
leader 1: dancing
7 processes created
```


3.8.4 Exclusive queue (3.8.4b.exclusive_queue.pml)

```
...
    19  proctype Leader(byte i) {
    20      wait(mutex)
    21      if
    22      :: followers > 0 ->
    23          followers--
    24          assert(followerQueue == 0)
    25          signal(followerQueue)
    26      :: else ->
    27          leaders++
    28          signal(mutex)
    29          wait(leaderQueue)
    30      fi
    31
    32  dance:
    33      printf("leader %d: to dance\n",i)
    34      wait(rendezvous)
    35      printf("leader %d: dancing\n",i)
    36      signal(mutex)
    37  }
    38
...
```

3.8.4 Exclusive queue (3.8.4b.exclusive_queue.pml)

```
...
    39  proctype Follower(byte i) {
    40      wait(mutex)
    41      if
    42      :: leaders > 0 ->
    43          leaders--
    44          assert(leaderQueue == 0)
    45          signal(leaderQueue)
    46      :: else ->
    47          followers++
    48          signal(mutex)
    49          wait(followerQueue)
    50      fi
    51
    52  dance:
    53      atomic {
    54          signal(rendezvous)
    55          printf("follower %d: dancing\n",i)
    56      }
    57  }
    58
...
```


3.8.4 Exclusive queue (3.8.4b.exclusive_queue.pml)

```
$ spin -run 3.8.4b.exclusive_queue.pml | expand

(Spin Version 6.4.6 -- 2 December 2016)
        + Partial Order Reduction

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

State-vector 68 byte, depth reached 77, errors: 0
...
```

3.8.4 Exclusive queue (3.8.4c.exclusive_queue.pml)

```
$ cat -n 3.8.4c.exclusive_queue.pml | expand

     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.8 Queue
     7      3.8.4 Exclusive queue solution
     8
     9      vk, 2017
    10  */
    11
    12  #include "Semaphore.h"
    13
    14  #define N 5
    15
    16  Semaphore mutex=1, leaderQueue=0, followerQueue=0, rendezvous=0
    17  byte      leaders=0, followers=0
    18
...
```


3.8.4 Exclusive queue (3.8.4c.exclusive_queue.pml)

```
$ spin -run 3.8.4c.exclusive_queue.pml | expand
pan:1: invalid end state (at depth 59)
pan: wrote 3.8.4c.exclusive_queue.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 60 byte, depth reached 60, errors: 1
...
```

3.8.4 Exclusive queue (3.8.4c.exclusive_queue.pml)

```
$ spin -run -E 3.8.4c.exclusive_queue.pml | expand

(Spin Version 6.4.6 -- 2 December 2016)
        + Partial Order Reduction

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

State-vector 60 byte, depth reached 60, errors: 0
...
```


3.8.4 Exclusive queue (3.8.4d.exclusive_queue.pml)

```
$ cat -n 3.8.4d.exclusive_queue.pml | expand

     1  /*  The Little Book of Semaphores (2.2.1)
     2      by A. Downey
     3
     4      Chapter 3. Basic synchronization patterns
     5
     6      3.8 Queue
     7      3.8.4 Exclusive queue solution
     8
     9      vk, 2017
    10  */
    11
    12  #include "Semaphore.h"
    13
    14  #define N 6
    15
    16  Semaphore mutex=1, leaderQueue=0, followerQueue=0, rendezvous=0
    17  byte      leaders=0, followers=0
    18
...
```

3.8.4 Exclusive queue (3.8.4d.exclusive_queue.pml)

```
...
    59  init {
    60      byte i
    61
    62      atomic {
    63          for (i: 0 .. N) {
    64              if
    65              :: i % 2 -> run Leader(i)
    66              :: else  -> run Follower(i)
    67              fi
    68          }
    69      }
    70  }
```


3.8.4 Exclusive queue (3.8.4d.exclusive_queue.pml)

```
$ spin -run 3.8.4d.exclusive_queue.pml | expand
pan:1: invalid end state (at depth 83)
pan: wrote 3.8.4d.exclusive_queue.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 76 byte, depth reached 84, errors: 1
...
```

3.8.4 Exclusive queue (3.8.4d.exclusive_queue.pml)

```
$ spin -run -E 3.8.4d.exclusive_queue.pml | expand

(Spin Version 6.4.6 -- 2 December 2016)
        + Partial Order Reduction

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

State-vector 76 byte, depth reached 84, errors: 0
...
```