MODULE *LZ*

EXTENDS *Integers*

```
CONSTANTS Processes,     The numbver of processes (threads actually) that can communicate. *\
          Signals,       The set of names of signal that can be sent by any thread. *\
          WorkingSet     Values that can be assigned at any working step *\
```

```
**    this is a comment containing the PlusCal code ***
--algorithm LZ{

 The only new variable is a set of locks
variables Locks = [p ∈ Signals ↦ 0],
          ProcValues = [p ∈ Processes ↦ 0],
          ProcStates = [p ∈ Processes ↦ "working"],
          SigStates = [s ∈ Signals ↦ "idle"],
          SigStorage = [s ∈ Signals ↦ 0] ;
          ProcSignals = [s ∈ Processes ↦ 0] ;

process ( p ∈ Processes )
{

    First, we decide to work or do some signal exchange
   i1: either
          goto w1 ;
       or
          goto c1 ;

    Work: change the process value
   w1: with ( i ∈ WorkingSet ) {
             ProcValues[self] := i ;
        } ;
        goto i1 ;

    Choose a signal before doing anything.
   c1: with ( s ∈ Signals ) {
            ProcSignals[self] := s ;
        } ;
       goto c2 ;

    Then send or receive a signal
   c2: either
           goto f1 ;
       or
           goto s1 ;

   Here we come to receive a signal. The first action in the case is flagging
```

```
$f1$: await $Locks[ProcSignals[self]] = 0$;
    $Locks[ProcSignals[self]] := 1$;
    goto $f2$;
```

Change both the signal state and process state. Note, the we will prove that it is a mutual exclusive section later.

```
$f2$: if ( $SigStates[ProcSignals[self]] =$ “idle” ) {
        $SigStates[ProcSignals[self]] :=$ “waiting” ;
        $ProcStates[self] :=$ “ready” ;
        goto $f3$;
     }
    else {
         Here we stop because the algorithm is intended for a model checker and
         not for a normal execution. This is the semantics of the “assume” statement.
        await FALSE;
     } ;
```

Release the lock and proceed to the signal receiving.

```
$f3$: $Locks[ProcSignals[self]] := 0$;
    goto $r1$;
```

Now we again do locking and become ready to getting values.

```
$r1$: await $Locks[ProcSignals[self]] = 0$;
    $Locks[ProcSignals[self]] := 1$;
    goto $r2$;
```

Receive values.

```
$r2$: if ( $SigStates[ProcSignals[self]] =$ “set” ) {
        $ProcValues[self] := SigStorage[ProcSignals[self]]$;
        $SigStates[ProcSignals[self]] :=$ “idle” ;
        $ProcStates[self] :=$ “working” ;
        goto $r5$;
     }
    else {
         Here we stop because the algorithm is intended for a model checker and
         not for a normal execution. This is the semantics of the “assume” statement.
        await FALSE;
     } ;
```

Release the lock and go to the initial process state.

```
$r5$: $Locks[ProcSignals[self]] := 0$;
    goto $i1$;
```

This action is sending one.

```
$s1$: await $Locks[ProcSignals[self]] = 0$;
    $Locks[ProcSignals[self]] := 1$;
    goto $s2$;
    Send values
  s2: if ( $SigStates[ProcSignals[self]] =$ "waiting" ) {
          $SigStorage[ProcSignals[self]] := ProcValues[self]$ ;
          $SigStates[ProcSignals[self]] :=$ "set" ;
          $ProcStates[self] :=$ "working" ;
          goto $s5$ ;
      }
      else {
          Here we stop because the algorithm is intended for a model checker and
          not for a normal execution. This is the semantics of the "assume" statement.
          await FALSE ;
      } ;

    Release the lock and then go to the very beginning.
  s5: $Locks[ProcSignals[self]] := 0$ ;
      goto $i1$ ;
}

}
```

**** this ends the comment containg the pluscal code *********

BEGIN TRANSLATION

VARIABLES $Locks, ProcValues, ProcStates, SigStates, SigStorage, ProcSignals, pc$

$vars \triangleq \langle Locks, ProcValues, ProcStates, SigStates, SigStorage, ProcSignals, pc \rangle$

$ProcSet \triangleq (Processes)$

$Init \triangleq$ Global variables

$$\begin{aligned}
&\land Locks = [p \in Signals \mapsto 0] \\
&\land ProcValues = [p \in Processes \mapsto 0] \\
&\land ProcStates = [p \in Processes \mapsto \text{"working"}] \\
&\land SigStates = [s \in Signals \mapsto \text{"idle"}] \\
&\land SigStorage = [s \in Signals \mapsto 0] \\
&\land ProcSignals = [s \in Processes \mapsto 0] \\
&\land pc = [self \in ProcSet \mapsto \text{"i1"}]
\end{aligned}$$

$$\begin{aligned}
i1(self) \triangleq\ &\land pc[self] = \text{"i1"} \\
&\land \lor \land pc' = [pc \text{ EXCEPT } ![self] = \text{"w1"}] \\
&\quad\ \lor \land pc' = [pc \text{ EXCEPT } ![self] = \text{"c1"}] \\
&\land \text{UNCHANGED } \langle Locks, ProcValues, ProcStates, SigStates, \\
&\qquad\qquad SigStorage, ProcSignals \rangle
\end{aligned}$$

$$\begin{aligned}
w1(self) \triangleq\ &\land pc[self] = \text{"w1"} \\
&\land \exists\, i \in WorkingSet :
\end{aligned}$$

```
            $ProcValues' = [ProcValues \text{ EXCEPT } ![self] = i]$
        $\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“i1”}]$
        $\land\ \text{UNCHANGED } \langle Locks, ProcStates, SigStates, SigStorage,$
                               $ProcSignals\rangle$

$c1(self) \triangleq$ $\land\ pc[self] = \text{“c1”}$
        $\land\ \exists\, s \in Signals :$
            $ProcSignals' = [ProcSignals \text{ EXCEPT } ![self] = s]$
        $\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“c2”}]$
        $\land\ \text{UNCHANGED } \langle Locks, ProcValues, ProcStates, SigStates,$
                               $SigStorage\rangle$

$c2(self) \triangleq$ $\land\ pc[self] = \text{“c2”}$
        $\land\ \lor\ \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“f1”}]$
          $\lor\ \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“s1”}]$
        $\land\ \text{UNCHANGED } \langle Locks, ProcValues, ProcStates, SigStates,$
                               $SigStorage, ProcSignals\rangle$

$f1(self) \triangleq$ $\land\ pc[self] = \text{“f1”}$
        $\land\ Locks[ProcSignals[self]] = 0$
        $\land\ Locks' = [Locks \text{ EXCEPT } ![ProcSignals[self]] = 1]$
        $\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“f2”}]$
        $\land\ \text{UNCHANGED } \langle ProcValues, ProcStates, SigStates, SigStorage,$
                               $ProcSignals\rangle$

$f2(self) \triangleq$ $\land\ pc[self] = \text{“f2”}$
        $\land\ \text{IF } SigStates[ProcSignals[self]] = \text{“idle”}$
             $\text{THEN }$ $\land\ SigStates' = [SigStates \text{ EXCEPT } ![ProcSignals[self]] = \text{“waiting”}]$
                  $\land\ ProcStates' = [ProcStates \text{ EXCEPT } ![self] = \text{“ready”}]$
                  $\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“f3”}]$
             $\text{ELSE }$ $\land\ \text{FALSE}$
                  $\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“f3”}]$
                  $\land\ \text{UNCHANGED } \langle ProcStates, SigStates\rangle$
        $\land\ \text{UNCHANGED } \langle Locks, ProcValues, SigStorage, ProcSignals\rangle$

$f3(self) \triangleq$ $\land\ pc[self] = \text{“f3”}$
        $\land\ Locks' = [Locks \text{ EXCEPT } ![ProcSignals[self]] = 0]$
        $\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“r1”}]$
        $\land\ \text{UNCHANGED } \langle ProcValues, ProcStates, SigStates, SigStorage,$
                               $ProcSignals\rangle$

$r1(self) \triangleq$ $\land\ pc[self] = \text{“r1”}$
        $\land\ Locks[ProcSignals[self]] = 0$
        $\land\ Locks' = [Locks \text{ EXCEPT } ![ProcSignals[self]] = 1]$
        $\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“r2”}]$
        $\land\ \text{UNCHANGED } \langle ProcValues, ProcStates, SigStates, SigStorage,$
                               $ProcSignals\rangle$
```

```
$r2(self) \triangleq \land pc[self] = \text{“r2”}$
    $\land \text{IF } SigStates[ProcSignals[self]] = \text{“set”}$
        $\text{THEN } \land ProcValues' = [ProcValues \text{ EXCEPT } ![self] = SigStorage[ProcSignals[self]]]$
             $\land SigStates' = [SigStates \text{ EXCEPT } ![ProcSignals[self]] = \text{“idle”}]$
             $\land ProcStates' = [ProcStates \text{ EXCEPT } ![self] = \text{“working”}]$
             $\land pc' = [pc \text{ EXCEPT } ![self] = \text{“r5”}]$
        $\text{ELSE } \land \text{FALSE}$
             $\land pc' = [pc \text{ EXCEPT } ![self] = \text{“r5”}]$
             $\land \text{UNCHANGED } \langle ProcValues, ProcStates, SigStates \rangle$
    $\land \text{UNCHANGED } \langle Locks, SigStorage, ProcSignals \rangle$

$r5(self) \triangleq \land pc[self] = \text{“r5”}$
    $\land Locks' = [Locks \text{ EXCEPT } ![ProcSignals[self]] = 0]$
    $\land pc' = [pc \text{ EXCEPT } ![self] = \text{“i1”}]$
    $\land \text{UNCHANGED } \langle ProcValues, ProcStates, SigStates, SigStorage,$
                      $ProcSignals \rangle$

$s1(self) \triangleq \land pc[self] = \text{“s1”}$
    $\land Locks[ProcSignals[self]] = 0$
    $\land Locks' = [Locks \text{ EXCEPT } ![ProcSignals[self]] = 1]$
    $\land pc' = [pc \text{ EXCEPT } ![self] = \text{“s2”}]$
    $\land \text{UNCHANGED } \langle ProcValues, ProcStates, SigStates, SigStorage,$
                      $ProcSignals \rangle$

$s2(self) \triangleq \land pc[self] = \text{“s2”}$
    $\land \text{IF } SigStates[ProcSignals[self]] = \text{“waiting”}$
        $\text{THEN } \land SigStorage' = [SigStorage \text{ EXCEPT } ![ProcSignals[self]] = ProcValues[self]]$
             $\land SigStates' = [SigStates \text{ EXCEPT } ![ProcSignals[self]] = \text{“set”}]$
             $\land ProcStates' = [ProcStates \text{ EXCEPT } ![self] = \text{“working”}]$
             $\land pc' = [pc \text{ EXCEPT } ![self] = \text{“s5”}]$
        $\text{ELSE } \land \text{FALSE}$
             $\land pc' = [pc \text{ EXCEPT } ![self] = \text{“s5”}]$
             $\land \text{UNCHANGED } \langle ProcStates, SigStates, SigStorage \rangle$
    $\land \text{UNCHANGED } \langle Locks, ProcValues, ProcSignals \rangle$

$s5(self) \triangleq \land pc[self] = \text{“s5”}$
    $\land Locks' = [Locks \text{ EXCEPT } ![ProcSignals[self]] = 0]$
    $\land pc' = [pc \text{ EXCEPT } ![self] = \text{“i1”}]$
    $\land \text{UNCHANGED } \langle ProcValues, ProcStates, SigStates, SigStorage,$
                      $ProcSignals \rangle$

$p(self) \triangleq i1(self) \lor w1(self) \lor c1(self) \lor c2(self) \lor f1(self)$
    $\lor f2(self) \lor f3(self) \lor r1(self) \lor r2(self) \lor r5(self)$
    $\lor s1(self) \lor s2(self) \lor s5(self)$
```

Allow infinite stuttering to prevent deadlock on termination.

```
$Terminating \triangleq \land \forall self \in ProcSet : pc[self] = \text{“Done”}$
```

$\land$ UNCHANGED *vars*

$$Next \triangleq (\exists\, self \in Processes : p(self)) \lor Terminating$$

$$Spec \triangleq Init \land \Box[Next]_{vars}$$

$$Termination \triangleq \Diamond(\forall\, self \in ProcSet : pc[self] = \text{“Done”})$$

END TRANSLATION

$EXT \triangleq$ INSTANCE ELZ

This is main result

THEOREM $Spec \Rightarrow EXT!Spec$

Additional checks of correctness

Two or more processes cannot be in a signal critical section at once. This condition is required because we do assignments at once in critical sections to simplify the refinement proof.

$$LockSafe \triangleq \Box\neg\exists\, p1 \in ProcSet,\ p2 \in ProcSet : \begin{array}{l} \land\, p1 \neq p2 \\ \land\, ProcSignals[p1] = ProcSignals[p2] \\ \land\, pc[p1] = pc[p2] \\ \land\, pc[p1] \in \{\text{“f2”}, \text{“r2”}, \text{“s2”}\} \end{array}$$

\* Modification History
\* Last modified *Mon Feb* 10 17:22:32 *MSK* 2020 by *zakharov*
\* Created *Fri Feb* 07 18:50:53 *MSK* 2020 by *zakharov*