

MODULE *OneBit*

This file contains the TLA+ representation of the One-Bit mutual exclusion algorithm *OB* of Section 4.2.5.2 of the book “A Science of Concurrent Programs” by *Leslie Lamport*. It contains the following:

- The TLA+ definition of formula *OB*.
- A *TLAPS* checked proof that the formula (4.19) of that section is an inductive invariant of *OB*.
- An outline of a TLA+ proof that *OB* satisfies the liveness property that is proved informally in Section 4.2.5.3.
- The TLA+ theorem asserting that *OB* implements the semaphore-based algorithm *LM*, as described in Section 6.4. The TLA+ formula *LM* is defined in module *LockMutex*.

All the theorems asserting that *OB* satisfies a property have been checked by *TLC*.

THE DEFINITION OF *OB*

We use the notation of Section 4.2.6.1 that the action describing execution of one step of process *p* beginning with control at a label such as *wait* is named *Wait(p)*. The value of *pc(p)* indicating control is at *wait* is written as the string “wait” in the TLA+ version. (We could declare *wait* to be a constant, but it’s more convenient to use strings so we don’t have to add assumptions such as that *wait* doesn’t equal *ncs* . As usual, *Init* and *Next* are the initial-state predicate and step predicate of the program.

EXTENDS *Integers*

VARIABLES *x, pc*

$v \triangleq \langle x, pc \rangle$

$Init \triangleq \wedge x = [p \in \{0, 1\} \mapsto \text{FALSE}]$   
 $\wedge pc = [p \in \{0, 1\} \mapsto \text{“ncs”}]$

$Ncs(p) \triangleq \wedge pc[p] = \text{“ncs”}$   
 $\wedge \text{TRUE}$   
 $\wedge pc' = [pc \text{ EXCEPT } ![p] = \text{“wait”}]$   
 $\wedge x' = x$

$Wait(p) \triangleq \wedge pc[p] = \text{“wait”}$   
 $\wedge x' = [x \text{ EXCEPT } ![p] = \text{TRUE}]$   
 $\wedge pc' = [pc \text{ EXCEPT } ![p] = \text{“w2”}]$

$W2(p) \triangleq \wedge pc[p] = \text{“w2”}$   
 $\wedge \text{IF } (p = 0)$   
 $\quad \text{THEN } \wedge \neg x[1]$   
 $\quad \wedge pc' = [pc \text{ EXCEPT } ![p] = \text{“cs”}]$   
 $\quad \text{ELSE } \wedge \text{IF } x[0]$   
 $\quad \quad \text{THEN } \wedge pc' = [pc \text{ EXCEPT } ![p] = \text{“w3”}]$   
 $\quad \quad \text{ELSE } \wedge pc' = [pc \text{ EXCEPT } ![p] = \text{“cs”}]$   
 $\wedge x' = x$

$W3(p) \triangleq \wedge pc[p] = \text{“w3”}$   
 $\wedge x' = [x \text{ EXCEPT } ![1] = \text{FALSE}]$

$$\begin{aligned}
& \wedge pc' = [pc \text{ EXCEPT } ![p] = \text{"w4"}] \\
W4(p) & \triangleq \wedge pc[p] = \text{"w4"} \\
& \wedge \neg x[0] \\
& \wedge pc' = [pc \text{ EXCEPT } ![p] = \text{"wait"}] \\
& \wedge x' = x \\
Cs(p) & \triangleq \wedge pc[p] = \text{"cs"} \\
& \wedge pc' = [pc \text{ EXCEPT } ![p] = \text{"exit"}] \\
& \wedge x' = x \\
Exit(p) & \triangleq \wedge pc[p] = \text{"exit"} \\
& \wedge x' = [x \text{ EXCEPT } ![p] = \text{FALSE}] \\
& \wedge pc' = [pc \text{ EXCEPT } ![p] = \text{"ncs"}] \\
PNext(p) & \triangleq Ncs(p) \vee Wait(p) \vee W2(p) \vee W3(p) \vee W4(p) \\
& \vee Cs(p) \vee Exit(p) \\
Next & \triangleq (\exists p \in \{0, 1\} : PNext(p)) \\
OB\text{Safe} & \triangleq Init \wedge \Box[Next]_v \\
OB\text{Fair} & \triangleq \forall p \in \{0, 1\} : WF_v((pc[p] \neq \text{"ncs"}) \wedge PNext(p)) \\
OB & \triangleq OB\text{Safe} \wedge OB\text{Fair}
\end{aligned}$$

---

VERIFYING THAT *OB* SATISFIES MUTUAL EXCLUSION

Here is the type-correctness invariant of program *OB*, defined in the book in (4.17).

$$\begin{aligned}
TypeOK & \triangleq \wedge x \in [\{0, 1\} \rightarrow \text{BOOLEAN}] \\
& \wedge pc \in [\{0, 1\} \rightarrow \{\text{"ncs"}, \text{"wait"}, \text{"w2"}, \text{"w3"}, \text{"w4"}, \text{"cs"}, \text{"exit"}\}] \\
& \wedge pc[0] \notin \{\text{"w3"}, \text{"w4"}\}
\end{aligned}$$

The following formula *InvUM* is the inductive invariant *Inv* of program *UM*, except with the type invariant *TypeOK* of *UM* replaced by the current definition of *TypeOK* for *OB*. It is an inductive invariant of *OB* that is strong enough to prove that *OB* satisfies mutual exclusion. The *TLC* model checker easily checks that *InvUM* is an invariant of *OB*, as well as condition *I2*, which that is the hard part of verifying inductive invariance.

*TLAPS* proof checker easily checks that *InvUN* implies the invariant *Mutex*, defined below, that expresses mutual exclusion.

$$\begin{aligned}
InvUM & \triangleq \wedge TypeOK \\
& \wedge \forall p \in \{0, 1\} : \wedge (pc[p] \in \{\text{"w2"}, \text{"cs"}\}) \Rightarrow x[p] \\
& \wedge (pc[p] = \text{"cs"}) \Rightarrow (pc[1-p] \neq \text{"cs"}) \\
Mutex & \triangleq \forall p, q \in \{0, 1\} : (p \neq q) \Rightarrow (\{pc[p], pc[q]\} \neq \{\text{"cs"}\})
\end{aligned}$$

Module *TLAPS* is needed when using the *TLAPS* proof checker.

INSTANCE *TLAPS*

THEOREM *InvUM*  $\Rightarrow$  *Mutex*

BY DEF *InvUM*, *TypeOK*, *Mutex*

Here are the definitions of *TypeOK* and the inductive invariant of *OB* given in formula (4.19) of the book, which we give the name *Inv*. As explained in the book, although the invariant *InvUM* proves that *OB* satisfies mutual exclusion, we need this stronger invariant *Inv* to prove liveness. *TLAPS* easily proves that the invariance of *Inv* implies mutual exclusion.

$$\begin{aligned} \text{Inv} &\triangleq \wedge \text{TypeOK} \\ &\wedge \forall p \in \{0, 1\} : (pc[p] = \text{"cs"}) \Rightarrow (pc[1 - p] \neq \text{"cs"}) \\ &\wedge x[0] \equiv (pc[0] \in \{\text{"w2"}, \text{"cs"}, \text{"exit"}\}) \\ &\wedge x[1] \equiv (pc[1] \in \{\text{"w2"}, \text{"w3"}, \text{"cs"}, \text{"exit"}\}) \end{aligned}$$

THEOREM  $\text{Inv} \Rightarrow \text{Mutex}$

BY DEF *Inv*, *TypeOK*, *Mutex*

Here is a *TLC* checked proof that *Inv* is an invariant of program *OB*. We need to give the theorem a name so it can be used in the proof of liveness. We give it the name *Invariance*

THEOREM  $\text{Invariance} \triangleq \text{OB} \Rightarrow \Box \text{Inv}$

In this proof,  $\langle 1 \rangle 1$  and  $\langle 1 \rangle 2$  are invariance conditions *I1* and *I2* of the book.

$\langle 1 \rangle 1$ .  $\text{Init} \Rightarrow \text{Inv}$

BY DEF *Init*, *Inv*, *TypeOK*

$\langle 1 \rangle 2$ .  $\text{Inv} \wedge [\text{Next}]_v \Rightarrow \text{Inv}'$

The SUFFICES step is the usual way of proving  $F \Rightarrow G$  by assuming  $F$  and proving  $G$ .

$\langle 2 \rangle$  SUFFICES ASSUME *Inv*,  
 $[\text{Next}]_v$   
 PROVE  $\text{Inv}'$

OBVIOUS

$\langle 2 \rangle$  USE DEF *Inv*, *TypeOK*

Since  $[\text{Next}]_v$  equals

$$\begin{aligned} &\vee \exists p \in \text{Procs} : \text{Ncs}(p) \\ &\vee \exists p \in \text{Procs} : \text{Wait}(p) \dots \\ &\vee \exists p \in \text{Procs} : \text{Exit}(p) \\ &\vee \text{UNCHANGED } v \end{aligned}$$

we can prove  $\text{Inv}'$  separately for each of the eight disjuncts. And we prove  $\exists p \in \{0, 1\} : A(p)$  by assuming  $p \in \{0, 1\}$  and proving  $A(p)$ , for each subaction  $A(p)$  of *Next*.

$\langle 2 \rangle 1$ . ASSUME NEW  $p \in \{0, 1\}$ ,

$\text{Ncs}(p)$

PROVE  $\text{Inv}'$

BY  $\langle 2 \rangle 1$  DEF *Ncs*

$\langle 2 \rangle 2$ . ASSUME NEW  $p \in \{0, 1\}$ ,

$\text{Wait}(p)$

PROVE  $\text{Inv}'$

BY  $\langle 2 \rangle 2$  DEF *Wait*

$\langle 2 \rangle 3$ . ASSUME NEW  $p \in \{0, 1\}$ ,

$\text{W2}(p)$

PROVE  $\text{Inv}'$

BY  $\langle 2 \rangle 3$  DEF  $W2$   
 $\langle 2 \rangle 4$ . ASSUME NEW  $p \in \{0, 1\}$ ,  
      $W3(p)$   
     PROVE  $Inv'$   
 BY  $\langle 2 \rangle 4$  DEF  $W3$   
 $\langle 2 \rangle 5$ . ASSUME NEW  $p \in \{0, 1\}$ ,  
      $W4(p)$   
     PROVE  $Inv'$   
 BY  $\langle 2 \rangle 5$  DEF  $W4$   
 $\langle 2 \rangle 6$ . ASSUME NEW  $p \in \{0, 1\}$ ,  
      $Cs(p)$   
     PROVE  $Inv'$   
 BY  $\langle 2 \rangle 6$  DEF  $Cs$   
 $\langle 2 \rangle 7$ . ASSUME NEW  $p \in \{0, 1\}$ ,  
      $Exit(p)$   
     PROVE  $Inv'$   
 BY  $\langle 2 \rangle 7$  DEF  $Exit$   
 $\langle 2 \rangle 8$ . CASE UNCHANGED  $v$   
 BY  $\langle 2 \rangle 8$  DEF  $v$   
 $\langle 2 \rangle 9$ . QED  
 BY  $\langle 2 \rangle 1, \langle 2 \rangle 2, \langle 2 \rangle 3, \langle 2 \rangle 4, \langle 2 \rangle 5, \langle 2 \rangle 6, \langle 2 \rangle 7, \langle 2 \rangle 8$  DEF  $Next, PNext$   
 $\langle 1 \rangle 3$ . QED  
 BY  $\langle 1 \rangle 1, \langle 1 \rangle 2, PTL$  DEF  $Init, Inv, OB, OBSafe$

*PTL* tells *TLAPS* to use a backend prover that reasons about propositional temporal logic.

---

#### VERIFYING THAT $OB$ SATISFIES ITS LIVENESS PROPERTY

---

The following theorem asserts that  $OB$  satisfies the liveness property (4.18) of the book. We outline here a TLA+ version of the informal proof in Section 4.2.5.3. *TLAPS* proofs are given for only the steps of the proof that don't actually depend upon the definition of the fairness property  $OBFair$  (except to show that it is a  $\Box$  property).

THEOREM  $OB \Rightarrow ((pc[0] \in \{\text{"wait"}, \text{"w2"}\}) \leadsto (pc[0] = \text{"cs"}))$

In temporal logic proofs, we need to reason with assumptions that are  $\Box$  formulas. So we have to replace  $Init$  in the definition of  $OB$  by a weaker  $\Box Inv$ . We will also have to convince *TLAPS* that  $OBFair$  is a  $\Box$  formula. We know that it's equivalent to  $\Box OBFair$ , but it suffices to prove that it's implied by  $OBFair$  and thus by  $OB$ .

$\langle 1 \rangle 1$ .  $OB \Rightarrow \Box Inv \wedge \Box [Next]_v \wedge \Box OBFair$

This is very easy to prove and *TLAPS* should be able to deduce it immediately from theorem *Invariance* and the definitions of  $OB$ ,  $OBSafe$ , and  $OBFair$ . However, currently it can't do it because its one back-end prover *PTL* that does temporal-logic reasoning doesn't know that  $\forall p \in S : \Box F(p)$  is equivalent to  $\Box (\forall p \in S : F(p))$ . *TLAPS* can prove it with a bit of help from us in this case because  $S$  equals  $\{0, 1\}$  and it has back-end provers that know  $\forall p \in \{0, 1\} : F(p)$  equals  $F(0) \vee F(1)$ .

$\langle 2 \rangle$  DEFINE  $Q(p) \triangleq WF_v((pc[p] \neq \text{"ncs"}) \wedge PNext(p))$

$\langle 2 \rangle 1$   $OBFair \equiv \forall p \in \{0, 1\} : Q(p)$

BY DEF  $OBFair$

$\langle 2 \rangle 2. Q(0) \wedge Q(1) \equiv \Box(Q(0) \wedge Q(1))$   
 BY *PTL*  
 $\langle 2 \rangle$  HIDE DEF  $Q$   
 $\langle 2 \rangle 3. Q(0) \wedge Q(1) \equiv (\forall p \in \{0, 1\} : Q(p))$   
 OBVIOUS  
 $\langle 2 \rangle 4. (\forall p \in \{0, 1\} : Q(p)) \equiv \Box(\forall p \in \{0, 1\} : Q(p))$   
 BY  $\langle 2 \rangle 2, \langle 2 \rangle 3, PTL$   
 $\langle 2 \rangle 5. OBFair \equiv \Box OBFair$   
 BY  $\langle 2 \rangle 1, \langle 2 \rangle 4, PTL$   
 $\langle 2 \rangle 6. QED$   
 BY  $\langle 2 \rangle 5, Invariance$  DEF  $OB, OBSafe$

The following step justifies the label on the box of the leads-to lattices of Figures 4.4 and 4.5. Because the step has no number (just a level  $\langle 1 \rangle$ ), its ASSUME formula is implicitly usable in all ensuing proofs.

$\langle 1 \rangle$  SUFFICES ASSUME  $\Box Inv \wedge \Box[Next]_v \wedge \Box OBFair$   
 PROVE  $(pc[0] \in \{\text{"wait"}, \text{"w2"}\}) \rightsquigarrow (pc[0] = \text{"cs"})$   
 BY  $\langle 1 \rangle 1$

Steps  $\langle 1 \rangle 2, \langle 1 \rangle 3$ , and  $\langle 1 \rangle 4$  correspond to the arcs numbered 1, 2, and 3 respectively in the leads-to lattice of Figure 4.4.

$\langle 1 \rangle 2 (pc[0] \in \{\text{"wait"}, \text{"w2"}\}) \Rightarrow (pc[0] = \text{"wait"}) \vee (pc[0] = \text{"w2"})$   
 OBVIOUS

$\langle 1 \rangle 3. (pc[0] = \text{"wait"}) \rightsquigarrow (pc[0] = \text{"w2"})$

$\langle 1 \rangle 4. (pc[0] = \text{"w2"}) \rightsquigarrow (pc[0] = \text{"cs"})$

The proof of  $\langle 1 \rangle 4$  is based on the proof lattice of Figure 4.5. However, step  $\langle 2 \rangle 1$  uses a little temporal logic to show that we can prove  $\langle 1 \rangle 4$  by erasing the arc from the top node  $pc[0] = \text{"w2"}$  and assuming  $\Box \neg (pc[0] = \text{"cs"})$ .

$\langle 2 \rangle 1. SUFFICES ASSUME \Box \neg (pc[0] = \text{"cs"})$   
 PROVE  $(pc[0] = \text{"w2"}) \rightsquigarrow (pc[0] = \text{"cs"})$   
 BY *PTL*

Steps  $\langle 2 \rangle 2$  and  $\langle 2 \rangle 3$  are the assertions made by the arcs 1 and 2, respectively, of the leads-to lattice of Figure 4.5 minus the arc removed in step  $\langle 2 \rangle 1$ .

$\langle 2 \rangle 2. (pc[0] = \text{"w2"}) \Rightarrow (pc[0] = \text{"w2"}) \wedge \Box(pc[0] \neq \text{"cs"})$   
 $\langle 3 \rangle \neg(pc[0] = \text{"cs"}) \equiv (pc[0] \neq \text{"cs"})$

We have to make this a separate step because the *PTL* backend prover can't figure it out by itself.

OBVIOUS

$\langle 3 \rangle QED$

BY  $\langle 2 \rangle 1, PTL$

$\langle 2 \rangle 3. (pc[0] = \text{"w2"}) \wedge \Box(pc[0] \neq \text{"cs"}) \Rightarrow \Box((pc[0] = \text{"w2"}) \wedge x[0])$

For convenience, we define a couple of abbreviations.

$\langle 2 \rangle$  DEFINE  $P1 \triangleq pc[1] \in \{\text{"cs"}, \text{"exit"}, \text{"ncs"}\}$   
 $P2 \triangleq pc[1] \in \{\text{"wait"}, \text{"w2"}, \text{"w3"}, \text{"w4"}\}$

Step  $\langle 2 \rangle 4$  says that having proved  $\langle 2 \rangle 2$  and  $\langle 2 \rangle 3$ , we can prove the goal introduced by  $\langle 2 \rangle 1$  by proving  $P1 \vee P2 \leadsto (pc[0] = \text{"cs"})$ . This is true because the assumption  $\Box Inv$  in the label of the outer box implies  $\Box(P1 \vee P2)$ .

$\langle 2 \rangle 4$ . SUFFICES ASSUME  $\Box(pc[0] = \text{"w2"}) \wedge \Box x[0]$   
PROVE  $P1 \vee P2 \leadsto (pc[0] = \text{"cs"})$   
 $\langle 3 \rangle Inv \Rightarrow P1 \vee P2$   
BY DEF *Inv*, *TypeOK*  
 $\langle 3 \rangle$  QED  
BY  $\langle 2 \rangle 2$ ,  $\langle 2 \rangle 3$ , *PTL*

The goal introduced by step  $\langle 2 \rangle 4$  is proved by the rest of the leads-to lattice of Figure 4.5. The assertions made by arcs of that part of the leads-to lattice are steps  $\langle 2 \rangle 5$  through  $\langle 2 \rangle 9$ .

$\langle 2 \rangle 5$ .  $P1 \leadsto (\Box(pc[1] = \text{"ncs"}) \vee P2)$  arcs 4  
 $\langle 2 \rangle 6$ .  $\Box(pc[1] = \text{"ncs"}) \leadsto \Box \neg x[1]$  arc 5  
 $\langle 2 \rangle 7$ .  $P2 \leadsto \Box(pc[1] = \text{"w4"})$  arc 6  
 $\langle 2 \rangle 8$ .  $\Box(pc[1] = \text{"w4"}) \leadsto \Box \neg x[1]$  arc 7  
 $\langle 2 \rangle 9$ .  $\Box \neg x[1] \leadsto (pc[0] = \text{"cs"})$  arc 8  
 $\langle 2 \rangle 10$ . QED

BY  $\langle 2 \rangle 5$ ,  $\langle 2 \rangle 6$ ,  $\langle 2 \rangle 7$ ,  $\langle 2 \rangle 8$ ,  $\langle 2 \rangle 9$ , *PTL*  
 $\langle 1 \rangle 5$ . QED  
BY  $\langle 1 \rangle 2$ ,  $\langle 1 \rangle 3$ ,  $\langle 1 \rangle 4$ , *PTL*

---

#### VERIFYING THAT *OB* IMPLEMENTS AN ALGORITHM USING LOCKS

Section 6.4 of the book shows that *OB* implements (refines) the simple mutual exclusion algorithm *LM* of Figure 4.6 in Section 4.2.6.1 when *LM* uses a weakly fair semaphore. *TLC* can automatically check the correctness of this result.

*OB* implements algorithm *LM* under the refinement mapping

$pc \leftarrow pcBar, sem \leftarrow semBar$

where *pcBar* and *semBar* are defined in Section 6.4.1 as follows:

$pcBar \triangleq [p \in \{0, 1\} \mapsto$   
IF  $pc[p] \in \{\text{"w2"}, \text{"w3"}, \text{"w4"}\}$  THEN "wait" ELSE  $pc[p]$   
 $semBar \triangleq$  IF  $\exists p \in \{0, 1\} : pc[p] \in \{\text{"cs"}, \text{"exit"}\}$  THEN 0 ELSE 1

*LM* is defined in module *LockMutex*. For every symbol *F* defined in that module, the following statement defines  $LM!F$  to have the definition obtained by applying the refinement mapping to the definition of *F* in *LockMutex*. If *F* has noarguments, this defines  $LM!F$  to equal:

$F$  WITH  $sem \leftarrow semBar, pc \leftarrow pcBar$

In particular, for *F* equal to *LM*, this statement defines  $LM!LM$  so that the meta-formula (6.8) can be written in this module as the theorem that follows the statement.

$LM \triangleq$  INSTANCE *LockMutex* WITH  $sem \leftarrow semBar, pc \leftarrow pcBar$

THEOREM  $OB \Rightarrow LM!LM$

Because the abstract program *OB* has so few reachable states, *TLC* can easily check that this theorem is true.

---

\\* Modification History  
\\* Last modified Sun *Oct* 20 16:34:12 *CEST* 2024 by *lamport*  
\\* Last modified *Tue Oct* 15 10:48:12 *CEST* 2024 by *lblam*  
\\* Created Sun *Jan* 22 15:11:24 *PST* 2023 by *lamport*