Technical Report
Verification of MCSHLi (whh596B)

Wim H. Hesselink, Peter A. Buhr, Ting-Ching Li

July 5, 2023

1 Introduction

This technical report supports the article [4] by verifying mutual exclusion, proper handling of pointers, and absence of deadlock. It has been developed with the proof assistant PVS [5]. The corresponding proof script can be obtained from [1].

1.1 Machines and threaded machines

The concurrent algorithm MCSHLi is modelled here as a threaded machine, as used in (e.g.) [2].

Recall that a *machine* or *state machine* is a tuple $K = (X, X_0, N)$ where X is a set, X_0 is a subset of X, and N is a reflexive binary relation on X. The elements of X are called *states*, X_0 is the *initialization*, N is the *next-state relation*.

An *execution* of machine K is a sequence of states that begins in the initialization and in which every pair of subsequent states satisfies the next-state relation. Formally, it is a function $xs : \mathbb{N} \to X$ such that $xs(0) \in X_0$ and that $(xs(n), xs(n+1)) \in N$ for all $n \in \mathbb{N}$.

A *predicate* is a Boolean function on the state space X. A predicate P can also be regarded as the subset of X where P holds. A predicate is called an *invariant* of machine K iff it contains all states of all executions of K.

A *threaded* machine has a set T of thread identifiers (natural numbers). Its next-state relation is a union $N = \mathbf{1}_X \cup \bigcup_{p \in T} N_p$, where $\mathbf{1}_X$ is the equality relation on X, and N_p is a next-state relation for thread p. The elements of N_p are regarded as steps that thread p can perform. So, apart from stuttering in $\mathbf{1}_X$, every step is done by some thread.

1.2 Invariants

We need a bit of theory concerning invariants and propose a method for obtaining and proving them.

A subrelation of the next-state relation N is called a *command*. For a command S and predicates P and Q, the *Hoare triple* $\{P\}S\{Q\}$ is the proposition that

$$\forall\, x, y : (x, y) \in S \ \wedge\ x \in P \ \Rightarrow\ y \in Q \text{ , or equivalently}$$
$$[P \ \Rightarrow\ \mathbf{wp}(S, Q)] \text{ ,}$$

where **wp** stands for Dijkstra's weakest precondition.

A predicate P is said to be *preserved* by command S iff $\{P\}S\{P\}$. Predicate P is called *stable* if it is preserved by the next-state relation N. A predicate is called *inductive* iff it is stable and holds initially. Every inductive predicate is an invariant.

A predicate P is said to be *threatened* by a command S iff it is not preserved by S. If predicate P is threatened by command S, a predicate Q is called a *remedy* for P and S iff $\{P \wedge Q\}S\{P\}$.

Let $\mathcal{C}$ be a set of commands such that $N = \mathbf{1}_X \cup \bigcup \mathcal{C}$. A family of predicates is called *complete* if any member of the family that is threatened by any command in $\mathcal{C}$, has some remedy consisting of members of the family. The conjunction of a complete family is stable. The family is said to be *initialized* if the initial condition implies every member of the family. The conjunction of an initialized complete family is an inductive predicate; every member of it is an invariant because it is implied by an invariant. This is the method used below to obtain and prove invariants. It is presumably well known, but it was first made explicit in [2].

For this paper, the proof assistant PVS [5] has been used to determine and verify the threatenings and the remedies for the invariants. We use predicates with names of the form *Xqd* for the ease of using query-replace in the PVS proof script. This script can be obtained from [1].

2 The correctness of the lock MCSHLi

In Section 2.1, the algorithm MCSHLi is modelled by a transition system. In Section 2.2, the method of Section 1.2 is used to generate a family of invariants that proves mutual exclusion. Section 2.3 proves that the algorithm is deadlock free.

2.1 Modelling MCSHLi for correctness

The starting point is the transition system of Figure 2 of [4], rendered here as Figure 1.

Recall that each line number stands for one atomic command. The implicit private variable pc_q indicates the line number that thread q is to execute next. To indicate which threads are where in the execution, we use the state-dependent sets of threads:

$$[k] = \{q \mid pc_q = k\} \ ,$$
$$[j,k] = \{q \mid j \le pc_q \le k\} \ .$$

The first aim is to prove mutual exclusion. As *CS* is at line 27, this is expressed by the predicate

MX0: $\quad q \in [27] \ \wedge \ r \in [27] \ \Rightarrow \ q = r \ .$

From this point onward, the predicates are given with implicit universal quantification over all free variables (here q and r).

2.2 Mutual exclusion for MCSHLi

In this section, the method of Section 1.2 is used to generate the invariants for mutual exclusion. The transition system has only 19 transitions. More than 40 invariants are needed to prove mutual exclusion.

For line number k and thread p, let $N_{p,k}$ be the command that corresponds to execution of line k by thread p. Let $\mathcal{C}$ be the set of all these commands. The idea is to construct a family of predicates such that every member of it that is threatened by some command in $\mathcal{C}$ has a remedy in the family. In most cases, the command is indicated by the line number, while the acting thread p is kept implicit.

Before proceeding into meaningful invariants, note that by construction it always holds that $my_q \ne \perp$ and that $1 \le \texttt{low}$ and $1 \le \texttt{high}$. These obvious invariants are used implicitly.

The first claim is that the ghost variables $slot_q$ and $\texttt{low}$ satisfy the invariants

Iq1: $\quad slot_q = slot_r \ne 0 \ \Rightarrow \ q = r \ ,$
Iq2: $\quad q \in M \ \Rightarrow \ slot_q = \texttt{low} \ ,$

where $M = M_1 \cup M_2 \cup M_3$ and

$$\begin{aligned} F \ &= \ \{q \mid (q \in [15] \wedge prev_q = \perp) \vee q \in [16]\} \ , \\ M_1 \ &= \ \{q \mid q \in F \ \wedge \ \texttt{flag}\} \ , \\ M_2 \ &= \ \{q \mid q \in [19] \ \wedge \ \neg\, \texttt{locked}(my_q)\} \ , \\ M_3 \ &= \ [17] \cup [20, 29] \ . \end{aligned}$$

It is easy to see that the predicates *Iq1* and *Iq2* together imply *MX0*. In fact, they imply the much stronger assertion

MX1: $\quad q \in M \ \wedge \ r \in M \ \Rightarrow \ q = r \ .$

initially:
$\mathtt{flag} = true \ \wedge\ \mathtt{tail} = \bot \ \wedge\ \mathtt{local} = \{\bot\}$
$\wedge\ \ \mathtt{low} = \mathtt{high} = 1$
$\wedge\ \ \forall\, q \in thread : pc_q = 11 \ \wedge\ slot_q = 0$.

loop of thread p :
11 NCS ;
choose $my_p \notin \mathtt{local}$; **add** my_p **to** $\mathtt{local}$;
12 $\mathtt{next}(my_p) := \bot$;
13 $\mathtt{locked}(my_p) := true$;
14 $prev_p := \mathtt{tail}$; $\mathtt{tail} := my_p$;
$slot_p := \mathtt{high}$; $\mathtt{high}$++ ;
15 **if** $prev_p = \bot$ **then**
16   **await** $(\mathtt{flag})$;
17     $\mathtt{flag} := false$
else
18     $\mathtt{next}(prev_p) := my_p$;
19   **await** $(\neg\,\mathtt{locked}(my_p))$
endif ;
20 $nxmy_p := succ_p := \mathtt{next}(my_p)$;
21 **if** $succ_p = \bot$ **then**
22   **if** $\mathtt{tail} = my_p$ **then** $\mathtt{tail} := \bot$
    else
23   **await** $(\mathtt{next}(my_p) \neq \bot)$;
24         $nxmy_p := succ_p := \mathtt{next}(my_p)$
    endif
endif ;
25 $\mathtt{mess} := succ_p$;
26 $succ_p := \bot$; **remove** my_p **from** $\mathtt{local}$;
27 CS ;
28 $succ_p := \mathtt{mess}$;
29 **if** $succ_p \neq \bot$ **then** $\mathtt{locked}(succ_p) := false$
else $\mathtt{flag} := true$ **endif** ;
$\mathtt{low}$++
endloop .

Figure 1: State machine of the lock MCSHLi, with ghost variables

We now take *Iq1* and *Iq2* as the founding members of an initialized complete family. This family is constructed in the following way. For each new member of the family, a list of line numbers of threatening commands is determined, with for each line number a remedy that is a conjunction of one or more, possibly new, members. As announced above, this analysis was performed with the proof assistant PVS [5], see the proof script in [1]. It turns out that 41 members are needed to make the family complete. All members hold initially.

The construction goes as follows. Predicate *Iq1* is threatened only by step 14. It has the remedy

Iq3: $\quad slot_q < \mathtt{high}$.

Predicate *Iq2* is threatened only by steps 14, 18, 29. At 14 and 18, it has the respective remedies

Iq4: $\quad \mathtt{tail} = \bot \ \wedge\ \mathtt{flag} \ \Rightarrow\ \mathtt{low} = \mathtt{high}$.
Iq5: $\quad q \in [14, 18] \ \Rightarrow\ \mathtt{locked}(my_q)$.

At step 29, it has as remedy the conjunction of *Iq1*, *Iq2*, and

Iq6: $\quad q \in F \ \wedge\ r \in [20, 29] \ \Rightarrow\ \mathtt{low} + 1 = slot_q$,
Iq7: $\quad q \in [21, 26] \ \vee\ q \in [29] \ \Rightarrow\ succ_q = nxmy_q$,
Iq8: $\quad q \in [21, 29] \ \wedge\ nxmy_q = my_r \ \wedge\ r \in [15, 26] \ \Rightarrow\ \mathtt{low} + 1 = slot_r$.

Predicate *Iq3* is inductive. Predicate *Iq4* is threatened only by the steps 22 and 29. It has the remedies *Jq1* at 22, and *Jq2* and *Jq3* at 29, where

Jq1: $\quad q \in [20, 29] \ \Rightarrow\ \neg\,\mathtt{flag}$.
Jq2: $\quad q \in [25, 29] \ \wedge\ \mathtt{tail} = \bot \ \Rightarrow\ nxmy_q = \bot$.
Jq3: $\quad q \in [25, 29] \ \wedge\ \mathtt{tail} = \bot \ \Rightarrow\ \mathtt{low} + 1 = \mathtt{high}$.

Predicate *Iq5* is threatened only by step 29. It has the remedies *Iq7* and

Jq4: $\quad q \in [21, 29] \ \wedge\ r \in [12, 18] \ \Rightarrow\ nxmy_q \neq my_r$.

Predicate *Iq6* is threatened only by the steps 14, 17, 19, 29. At step 14, it has the remedies *Jq3* and

Jq5: $\quad q \in [15, 24] \ \Rightarrow\ \mathtt{tail} \neq \bot$.

At step 29, it has the remedy *MX1*. At the steps 17 and 19, it has the respective remedies

Jq6: $\quad q \in F \ \wedge\ r \in [17] \ \Rightarrow\ slot_q = \mathtt{low} + 1$,
Jq7: $\quad q \in F \ \wedge\ r \in [19] \ \wedge\ \neg\,\mathtt{locked}(my_r) \ \Rightarrow\ slot_q = \mathtt{low} + 1$.

Predicate *Iq7* is threatened only by step 28. It has the remedy

Jq8: $\quad q \in [26, 28] \ \Rightarrow\ \mathtt{mess} = nxmy_q$.

Predicate *Iq8* is threatened only by steps 14, 20, 24, and 29. At steps 14 and 29, it has the respective remedies *Jq4* and *MX1*. At steps 20 and 24, it has as remedy the conjunction of *Iq2* and

Jq9: $\quad q \in [15, 26] \ \wedge\ r \in [15, 26] \ \wedge\ \mathtt{next}(my_q) = my_r \ \Rightarrow\ slot_q + 1 = slot_r$.

Note that this predicate shows that the waiting threads form a queue numbered by *slot*.

Predicate *Jq1* is threatened only by the steps 19 and 29. At 29 it has the remedy *MX1*, at 19 the remedy

Kq1: $\quad q \in [19] \ \wedge\ \neg\,\mathtt{locked}(my_q) \ \Rightarrow\ \neg\,\mathtt{flag}$.

Predicate *Jq2* is threatened only by the steps 21, 22, 24. At 21 and 24, it has the remedy *Jq5*. At 22, it has the remedies *MX1* and

Kq2: $\quad q \in [21, 26] \ \wedge\ my_q = \mathtt{tail} \ \Rightarrow\ nxmy_q = \bot$.

Predicate *Jq3* is threatened only by the steps 21, 22, 24, 29. At 21 and 24, it has the remedy *Jq5*. At 29, it has the remedy *MX1*. At 22, it has the remedies *Iq2* and

Kq3: $q \in [15, 26] \ \wedge \ my_q = \texttt{tail} \ \Rightarrow \ slot_q + 1 = \texttt{high}$.

Predicate *Jq4* is threatened only by the steps 11, 20, 24. At 11, it has the remedy *Kq4*, at 20 and 24 the remedy *Kq5*, where

Kq4: $q \in [21, 29] \ \Rightarrow \ nxmy_q \in \texttt{local}$,
Kq5: $q \in [15, 26] \ \wedge \ r \in [12, 26] \ \wedge \ \texttt{next}(my_q) = my_r \ \Rightarrow \ r \in [19]$.

Predicate *Jq5* is threatened only by step 22. It has the remedy *Iq1*, *Iq2*, *Iq3*, *Kq3*, and

Kq6: $q \in [15, 29] \ \Rightarrow \ \texttt{low} \leq slot_q$.

Together with *Iq3*, this predicate implies that the slots are bounded by

$$q \in [15, 29] \ \Rightarrow \ \texttt{low} \leq slot_q < \texttt{high} \ .$$

Predicate *Jq6* is threatened only by the steps 14, 16, 29. It has the remedies *Jq5* at 14 and *MX1* at 29. At 16, it has the remedy

Kq7: $q \in F \ \wedge \ r \in F \ \Rightarrow \ q = r$.

Predicate *Jq7* is threatened only by the steps 14, 18, 29. It has the remedies *Jq5* at 14 and *Iq5* at 18. At 29, it has the remedies *MX1*, *Iq1*, *Iq6*, *Iq7*, *Iq8*.

Predicate *Jq8* is threatened only by step 25. It has the remedies *MX1* and *Iq7*.

Predicate *Jq9* is threatened only by the steps 14 and 18. At step 14, it has the remedies *Kq5* and

Kq8: $q \in [13, 14] \ \Rightarrow \ \texttt{next}(my_q) = \bot$.

At step 18, it has the remedies

Lq1: $q \in [12, 26] \ \wedge \ r \in [12, 26] \ \wedge \ my_q = my_r \ \Rightarrow \ q = r$,
Lq2: $q \in [15, 26] \ \wedge \ r \in [15, 18] \ \wedge \ my_q = prev_r \ \Rightarrow \ slot_q + 1 = slot_r$.

Predicate *Kq1* is threatened only by the steps 18 and 29. At step 18, it has the remedy *Iq5*, at step 29 the remedies *MX1* and *Jq1*.

Predicate *Kq2* is threatened only by the steps 14, 20, 24. At step 14, it has the remedy *Lq1*, at steps 20 and 24 the remedy

Lq3: $\texttt{tail} = \bot \ \vee \ \texttt{next}(\texttt{tail}) = \bot$.

Predicate *Kq3* is threatened only by step 14. It has the remedy *Lq1*.

Predicate *Kq4* is threatened only by the steps 20, 24, 26. At step 26, it has the remedies *Iq2* and *Iq8*. At steps 20 and 24, it has the remedy

Lq4: $q \in [15, 26] \ \Rightarrow \ \texttt{next}(my_q) \in \texttt{local}$.

Predicate *Kq5* is threatened only by the steps 11, 14, 18, 19. At step 11, it has the remedy *Lq4*, at step 14 the remedy *Kq8*, at step 18 the remedy *Lq1*, at step 19 the remedies *Iq1*, *Iq2*, *Jq9*, *Kq6*.

Predicate *Kq6* is threatened only by the steps 14 and 29. At step 14, it has the remedy *Lq5*, at step 29 the remedies *Iq1*, *Iq2*, and *Lq5*, where

Lq5: $\texttt{low} \leq \texttt{high}$.

Predicate *Kq7* is threatened only by step 14. It has the remedy *Jq5*.

Predicate *Kq8* is threatened only by step 18. It has the remedy

Lq6: $q \in [12, 14] \ \wedge \ r \in [15, 18] \ \Rightarrow \ my_q \neq prev_r$.

Predicate *Lq1* is threatened only by step 11. It has the remedy

Lq7: $q \in [12, 26] \;\Rightarrow\; my_q \in \texttt{local}$.

Predicate *Lq2* is threatened only by step 14. It has the remedies *Kq3*, *Lq6*, and

Lq8: $q \in [12, 14] \;\Rightarrow\; my_q \neq \texttt{tail}$.

Predicate *Lq3* is threatened only by the steps 14 and 18. At step 14, it has the remedy *Kq8*. At step 18, it has the remedy

Mq1: $q \in [15, 18] \;\Rightarrow\; prev_q \neq \texttt{tail}$.

Predicate *Lq4* is threatened only by the steps 12, 14, 18, 26. At step 12, it has the remedy

Mq2: $\perp \in \texttt{local}$.

At step 14, it has the remedies *Kq8* and *Mq2*, at step 18 the remedy *Lq7*, at step 26 the remedies *Iq2*, *Jq9*, *Kq6*.

Predicate *Lq5* is threatened only by step 29. It has the remedies *Iq2* and *Iq3*.

Predicate *Lq6* is threatened only by the steps 11 and 14. At step 14, it has the remedy *Lq8*, at step 11 the remedy

Mq3: $q \in [15, 18] \;\Rightarrow\; prev_q \in \texttt{local}$.

Predicate *Lq7* is threatened only by step 26. It has the remedy *Lq1*.

Predicate *Lq8* is threatened only by the steps 11 and 14. At step 14, it has the remedy *Mq1*, at step 11 the remedy

Mq4: $\texttt{tail} \in \texttt{local}$.

Predicate *Mq1* is threatened only by the steps 14 and 22. At step 14, it has the remedies *Lq6* and *Lq8*, at step 22 the remedies *Iq1*, *Iq2*, *Iq3*, *Kq3*, *Kq6*.

Predicate *Mq2* is inductive.

Predicate *Mq3* is threatened only by the steps 14 and 26. At step 14, it has the remedy *Mq4*, at step 26 the remedy

Mq5: $q \in [24, 26] \;\wedge\; r \in [15, 18] \;\Rightarrow\; my_q \neq prev_r$.

Predicate *Mq4* is threatened only by the steps 14, 22, and 26. At step 14, it has the remedy *Lq7*, at step 22 the remedy *Mq2*, at step 26 the remedy

Mq6: $q \in [23, 26] \;\Rightarrow\; my_q \neq \texttt{tail}$.

Predicate *Mq5* is threatened only by the steps 14, 21, 22, and 23. At step 14, it has the remedy *Mq6*, at step 21 the remedies *Mq2* and *Mq7*, at step 22 the remedy *Mq1*, at step 23 the remedy *Mq8*, where

Mq7: $q \in [21, 26] \;\wedge\; r \in [15, 18] \;\wedge\; my_q = prev_r \;\Rightarrow\; nxmy_q = \perp$.
Mq8: $q \in [15, 26] \;\wedge\; r \in [15, 18] \;\wedge\; my_q = prev_r \;\Rightarrow\; \texttt{next}(my_q) = \perp$.

Predicate *Mq6* is threatened only by the steps 14 and 21. At step 14, it has the remedy *Lq1*, at step 21 the remedies *Iq7* and *Kq2*.

Predicate *Mq7* is threatened only by the steps 14, 20, and 24. At step 14, it has the remedy *Kq2*, at steps 20 and 24 the remedy *Mq8*.

Predicate *Mq8* is threatened only by the steps 14 and 18. At step 14, it has the remedies *Kq8* and *Lq3*, at step 18 the remedies *Iq1* and *Lq2*.

This concludes the construction of a complete family of predicates that contains *Iq1* and *Iq2*, and hence implies *MX1*. It thus concludes the proof of mutual exclusion.

Almost all these invariants are (almost) equal to invariants used in [3]. The arguments used in the proofs are also very similar.

We also have the proof obligation that all pointers are in `local` when referred to. This is indeed captured in the invariants *Kq4*, *Lq4*, *Lq7*, *Mq2*, *Mq3*, *Mq4*.

2.3 No deadlock states

A thread is said to be *competing* if it is not at line 11. A state is called a *deadlock state* if there are competing threads, and none of them can do a step, i.e. execute a command. As the algorithm has no internal loops, deadlock-freedom is equivalent to the absence of deadlock states.

The proof of deadlock freedom needs two invariants with an existential quantification:

ExM: $\mathtt{low} < \mathtt{high} \;\Rightarrow\; \exists\, q : q \in M$,
Nq1: $q \in [15,24] \;\wedge\; \mathtt{next}(my_q) = \bot \;\wedge\; \mathtt{tail} \neq my_q$
$\Rightarrow\;\; \exists\, r : r \in [15] \cup [18] \;\wedge\; my_q = prev_r$.

Before proving these invariants, they are applied to prove absence of deadlock.

Theorem 1 *Assume there are competing threads. Then some competing thread can do a step.*

Proof. Every thread that is not at an **await** statement can do the step of its line number. We may therefore assume that every competing thread is at an **await** statement, i.e., at one of the lines 16, 19, 23. As there are competing threads, there is a thread at one of the lines 16, 19, 23. By *Iq3* and *Kq6*, it follows that $\mathtt{low} < \mathtt{high}$. The invariant *ExM* therefore gives the existence of some thread $q \in M$. Therefore thread q is at one of the lines 16, 19, 23. If q is at line 16 or 19, the definition of M implies that thread q can do the corresponding step. Therefore thread q is at line 23. If $\mathtt{next}(my_q) \neq \bot$, thread q can do the corresponding step. Otherwise, by *Mq6* and *Nq1*, there is a thread at line 15 or line 18. This contradicts the above assumption. □

In order to prove the invariants *ExM* and *Nq1*, one first observes that predicate *ExM* is logically implied by the predicates *Iq4* and

Nq2: $\mathtt{flag} \;\vee\; \exists\, q : (q \in [19] \;\wedge\; \neg\,\mathtt{locked}(my_q)) \;\vee\; q \in [20,29]$,
Nq3: $\mathtt{flag} \;\wedge\; \mathtt{tail} \neq \bot \;\Rightarrow\; \exists\, q : (q \in [15] \;\wedge\; prev_q = \bot) \;\vee\; q \in [16,17]$.

The invariant *Nq3* is the main difference between MCSHLi and MCSH: combined with some other invariants, it implies that the `flag` only holds when the queue is empty or has its head in $[15,17]$. In MCSH, the `flag` holds more often.

The set `local` of the meaningful pointers satisfies the inductive invariant

Nq4: $u \in \mathtt{local} \;\Rightarrow\; u = \bot \;\vee\; (\exists\, q : u = my_q \;\wedge\; q \in [12,26])$.

This invariant is used to derive the following existential version of *Iq8*:

Iq8E: $q \in [21,29] \;\wedge\; nxmy_q \neq \bot$
$\Rightarrow\;\; \exists\, r \in [19] : nxmy_q = my_r \;\wedge\; slot_r = \mathtt{low} + 1$.

Indeed, the combination of *Kq4* and *Nq4* gives a thread $r \in [12,26]$ with $nxmy_q = my_r$. Then one applies *MX1*, *Iq2*, *Iq8*, and *Jq4*.

After this preparation, the invariants *Nq1, Nq2*, *Nq3* can be proved. Predicate *Nq1* is threatened only by the steps 12 and 22. In both cases, *Lq1* serves as a remedy.

Predicate *Nq2* is threatened only by the steps 13 and 29. At step 13, it has the remedy *Lq1*. At step 29, it has the remedies *MX1*, *Iq7*, and *Iq8E*.

Predicate *Nq3* is threatened only by step 29. It has the remedies *Iq7* and

Nq5: $q \in [25,29] \;\wedge\; nxmy_q = \bot \;\wedge\; \mathtt{tail} \neq \bot$
$\Rightarrow\;\; \exists\, r : r \in F \;\wedge\; slot_r = \mathtt{low} + 1$.

Predicate *Nq5* is threatened only by the steps 14, 16, 21, 24, and 29. It has the respective remedies *Jq3*, *Jq1*, *Iq7*, *Nq6*, *MX1*, where

Nq6: $q \in [24] \;\Rightarrow\; \mathtt{next}(my_q) \neq \bot$.

Predicate *Nq6* is threatened only by step 12. It has the remedy *Lq1*. This completes the proof of the invariants *ExM* and *Nq1*, and thus the proof of deadlock freedom.

References

[1] W. H. Hesselink. Verification of hardware locks. `http://wimhesselink.nl/mechver/HardwareLocks`, 2021.

[2] W. H. Hesselink. Trylock, a case for temporal logic and eternity variables. *Science of Computer Programming*, 216(102767), 2022.

[3] W. H. Hesselink and P. A. Buhr. MCSH, a lock with the standard interface. *ACM Transactions on Parallel Computing*, 10:1–23, 2023.

[4] W. H. Hesselink, P. A. Buhr, and Ting-Ching Li. MCSHLi, modified MCSH lock with the standard interface. In preparation, 2023.

[5] S. Owre, N. Shankar, J.M. Rushby, and D.W.J. Stringer-Calvert. *PVS Version 7.1, System Guide, Prover Guide, PVS Language Reference*, 2020. `http://pvs.csl.sri.com`, accessed 1 Dec. 2021.