A Correction on “A Family of 2-process Mutual Exclusion Algorithms:
Notes on UNITY: 13-90”

Notes on UNITY: 22-90

Angela Dappert-Farquhar
University of Texas, Austin

12/90

On page 6 of the above note, \( I_1. [1 \leq m \leq 3 \equiv u] \land [3 \leq m \leq 4 \Rightarrow \neg p] \) and
\( I_2. [1 \leq n \leq 3 \equiv v] \land [3 \leq n \leq 4 \Rightarrow p] \)
are listed as invariants of the program 2-mutex, reproduced below.

Program 2-mutex

initially \( u, v, m, n = false, false, 0, 0 \)

assign
\{process u’s program\}
\( u, m := true, 1 \) if \( u.h \land m = 0 \)
\( p, m := v, 2 \) if \( m = 1 \)
\( m := 3 \) if \( \neg p \land m = 2 \)
\( u, m := false, 4 \) if \( m = 3 \)
\( p, m := true, 0 \) if \( m = 4 \)

\{process v’s program\}
\( v, n := true, 1 \) if \( v.h \land n = 0 \)
\( p, n := \neg u, 2 \) if \( n = 1 \)
\( n := 3 \) if \( p \land n = 2 \)
\( v, n := false, 4 \) if \( n = 3 \)
\( p, n := false, 0 \) if \( n = 4 \)

end.

If \( m = 4 \) then \( I_1 \) implies \( \neg u \land \neg p \).

In the possible execution \( \{m, n, u = 4, 1, false\} p, n := \neg u, 2 \) if \( n = 1 \) \( \{m, p = 4, true\} \),
the postcondition violates the conjunct \( [3 \leq m \leq 4 \Rightarrow \neg p] \) of \( I_1 \).

Analogously, for \( n = 4 \) and statement \( p, m := v, 2 \) if \( m = 1 \) the conjunct \( [3 \leq n \leq 4 \Rightarrow p] \)
of \( I_2 \) can be violated.

These invariants can be modified as follows
\( I_1. [1 \leq m \leq 3 \equiv u] \land [m = 3 \Rightarrow \neg p] \)
\( I_2. [1 \leq n \leq 3 \equiv v] \land [n = 3 \Rightarrow p] \).
The proofs of the safety and progress properties are not affected by this change.