COMPSYS 705 – Formal Methods For Engineers

Assigment – 2

Submission Date: **23/Oct/2022 (no extensions)**
Submission Format: 1 file called <your-upi>.tgz containing the following:

a.) A promela file, including the LTL formulas for Q1.

b.) A python script encoding a working SMT formulation for Q2.

c.) A pdf file/report explaining the results obtained from SPIN/SMT solvers for Q1 and Q2.

NOTE: Your code should be well commented

Q1.) This question relates to your understanding of model-checking LTL properties on concurrent processes.

Part-A
Model Petersons mutual exclusion algorithm as described below in Promela.

The basic idea behind Peterson's n-process mutual exclusion algorithm is that each process passes through $n-1$ stages before entering the critical section (cs). These stages are designed to block one process per stage so that after $n-1$ stages only one process will be eligible to enter the critical section (which we consider as stage n). The algorithm uses two integer arrays *step* and *pos* of sizes $n-1$ and n respectively: *pos* is an array of 1-writer multi-reader variables and *step* is an array of multi-writer multi-reader variables. The value at *step*[j] indicates the *latest* process at step j, and *pos*[i] indicates the latest stage that the process i is passing through. (Peterson uses Q for *pos*, and *TURN* for *step*.) The array *pos* is initialized to 0. The process id's, *pid*s, are assumed to be integers between 1 and n. The code segment for process i is given in Figure 1.

```
Process i:

for j = 1 to n − 1 do
begin
  pos[i] := j;
  step[j] := i;
  wait until (∀k ≠ i, pos[k] < j)
                 ∨(step[j] ≠ i)
end;
cs.i;
pos[i] := 0;
```

Figure – 1

Part-B
Represent the following properties in LTL and verify them against at least 2 processes from above.

Property-1 (Safety property): Multiple processes cannot enter the ciritical section together.
Property-2 (Liveness property): If a process is waiting, eventually it will enter the critical section.
Property-3 (Liveness property): Any process not in the critical section will eventually enter the critical

section.

Q2.) This question relates to your understanding of using SMT solvers for hardware verification.

Majority voter is a protocol used in fault tolerant systems. Consider 3-processors A, B, and C, carrying out the same computation simultaneously. Any of these processors might suffer from transient faults during processing. In the majority voter protocol, an output Y is set depending upon the majority result produced from the processors. The truth table below describes the majority voter protocol:

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The boolean equation:

$$Y = (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge B \wedge C) \text{ – (1)}$$

gives the functional description of the truth-table above. A hardware engineer states that he/she will implement the above circuit using the equation below:

$$Y' = (A \wedge B) \vee (B \wedge C) \vee (A \wedge C) \text{ – (2)}$$

Prove using the SMT solver that Equations (1) and (2) are equivalent. If they are not, show why not?