

Assignment 1

Total marks: 40.

Out: Feb 14

Due: Feb 27 by 17:00

Note: Your report for this assignment should be the result of your own individual work. Take care to avoid plagiarism (“copying”). You may discuss the problems with other students, but do not take written notes during these discussions, and do not share your written solutions.

1. [20 points] **Mutual Exclusion**

The following is a *Spin* description of a mutual exclusion algorithm using a semaphore variable "r". The algorithm is safe (does not allow two processes in the critical area at the same time), and does not deadlock. However, it is not fair (i.e. there is no guarantee, under weak fairness, that processes will eventually get into the critical region).

```
/* Mutex via semaphores
[]<>p1 will fail even with weak fairness turned on
*/

int r = 1; /* semaphore */
bit p1, p2 = 0;

active proctype P1() {
do
::   atomic{r > 0 -> r = 0}; /* request(r)*/
    p1 = 1; /* critical section */
    p1 = 0;
    atomic{r = 1}; /*release(r) */
od
}

active proctype P2() {
do
::
    atomic{r > 0 -> r = 0}; /* request(r)*/
    p2 = 1; /* critical section */
```

```

    p2 = 0;
    atomic{r = 1}; /*release(r)*/
od
}

```

Your task is to come up with an Event-B model for the above program. Prove that your model is safe (two processes cannot be in the critical section at the same time) and deadlock free.

Note that In Event-B you have to think in terms of events (such as enter or exit the critical region) rather than statements in a coding language. Make sure that you capture the requirements via invariants (and document these requirements via comments). Discharge all proof obligations.

Note also that you will need invariants in addition to those from the requirements. The additional invariants will capture which states are reachable from the initial state. This assignment should indicate the importance of invariants in developing correct concurrent software.

When you are finished, export your completed model as `mutex.zip` and then submit it as:

```
submit -l 3342 asg1 mutex.zip
```

2. [20 points] Parity Theorem

In the distributed FTP development discussed in class, we introduced a parity bit in the transmission protocol for efficiency. The parity function is defined in three axioms, from which we would like to prove the theorem “thm1”:

$$\text{axm1: } \textit{parity} \in \mathbb{N} \rightarrow \mathbb{N}$$

$$\text{axm2: } \textit{parity}(0) = 0$$

$$\text{axm3: } \forall x. x \in \mathbb{N} \Rightarrow \textit{parity}(x + 1) = 1 - \textit{parity}(x)$$

$$\text{thm1: } \forall x, y. x \in \mathbb{N} \wedge y \in \mathbb{N} \wedge x \in y..y + 1 \wedge \\ \textit{parity}(x) = \textit{parity}(y) \Rightarrow x = y.$$

You must enter the above axioms and theorem in a Rodin context and prove that the theorem is correct. You may add lemmas to facilitate the proof of theorem “thm1”, but you must then treat these lemmas as theorems and prove them as well.

When you are finished, export your completed model as `parity.zip` and then submit it as:

```
submit -l 3342 asg1 parity.zip
```