

# Event-B Labs 2&3 – Winter 2013

---

## 1. 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
}
```

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

**Hint:** In Event-B you have to think *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.

**Hint:** 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.

**Latex.** Rodin has a *Latex* plugin. To generate a Latex view of your machine (say "m"), right click on the machine and select the Latex menu. This will put a file "m.tex" in the Latex folder of your project (in the Eclipse workspace). From a command prompt on our Prism linux systems, you can run **pdflatex m.tex** and that will produce "m.pdf".

**Zip Export:** Export your Rodin project as "mutex.zip" (which saves the machine and all your proofs). We can then import your project and check your proofs. You can also use the Latex plugin to obtain a *pdf* printout of your machine "mutex.pdf".

## 2. Context: Nand Gate

A NAND gate can be constructed by inverting an AND gate. The following shows how to define an AND gate and inverter INV.

```
CONTEXT C0
CONSTANTS
  D    D = { 0,1}
  and  AND gate
  inv  Inverter
  nand  NAND gate specification
AXIOMS
  axm1 : D = {0, 1}
  axm2 : and ∈ D × D → D
  axm3 : inv ∈ D → D
  axm4 : (∀x, y. x ∈ D ∧ y ∈ D ⇒ (and(x ↦ y) = 1 ⇔ x = 1 ∧ y = 1))
          definition of AND gate
  axm5 : (∀x. x ∈ D ⇒ ((x = 1 ⇔ inv(x) = 0) ∧ (x = 0 ⇔ inv(x) = 1)))
          definition of inverter
```

You must now specify the behaviour of a NAND gate and use the definition to show that you can implement a NAND gate by inverting an AND gate.

Can you think of a nicer way to describe an inverter, than what is shown above?

## 3. Context: Set theory and predicate logic

Consider the following argument:

All birds have wings. No fish have wings. Therefore, no fish are birds.

Let  $W$  be the set of animals that have wings,  $B$  be the set of birds,  $F$  be the set of fishes. The above argument can be formalized as follows:

```
H1  B ⊆ W
H2  F ∩ W ⊆ ∅
⊢
G   F ∩ B ⊆ ∅
```

Formalize and prove the above in a Rodin **Context**.

## 4. Refinement and Witness: Celebrity

See the **celebrity** problem in the Rodin User's Handbook Section 2.9.1 (p. 61 to 71). The handbook is in the SVN under *docs*. Import *celebrity.zip* into your workspace to get started.<sup>1</sup> Work your way through this example until the final refinement.

---

<sup>1</sup> <http://handbook.event-b.org/current/files/Celebrity.zip>