CS 476 Homework #12 Due 10:45am on 11/17

Note: Answers to the exercises listed below and all Maude code should be emailed to `nishant2@illinois.edu`.

1. Recall that, since the readers-and-writers protocol specification discussed in Lecture 18 (slightly modified to avoid use of the built-in module `NAT`),

```
set include BOOL off .

mod R&W is
  sorts Nat Config .
  op 0 : -> Nat [ctor] .
  op s : Nat -> Nat [ctor] .
  op <_,_> : Nat Nat -> Config [ctor] .  --- readers/writers

  vars R W : Nat .

  rl < 0, 0 > => < 0, s(0) > .
  rl < R, s(W) > => < R, W > .
  rl < R, 0 > => < s(R), 0 > .
  rl < s(R), W > => < R, W > .
endm
```

   is infinite-state, in Lecture 18 we were only able to perform *bounded model checking* of the three invariants desired for this module, namely: (i) *mutual exclusion*, (ii) *one writer*, and (iii) *deadlock freedom*. In this problem you are asked to verify invariants (i)–(ii) by narrowing-based symbolic model checking. Problem 2 will deal with the symbolic verification of the *deadlock freedom* invariant.

   However, symbolically verifying invariants (i)–(ii) is not entirely trivial for two reasons: (a) the initial state is `< 0 , 0>` and it is not entirely clear how to generalize it to a symbolic initial state, and (ii) as remarked in pg. 14 of Lecture 20, *for ground terms narrowing coincides with rewriting*. That is, from `< 0 , 0>` narrowing search would just be performing a search equivalent to the standard `search` command in Maude, which can only be used effectively in a bounded search manner.

   Hint. There is however an alternative possibility to verify invariants (i)–(ii) by narrowing-based symbolic model checking, namely, by the method described in Appendix 2 to Lecture 20.

   Caveats: (1) The Maude command to use in order to perform narrowing with folding of the narrowing tree into a graph, is the `fvu-narrow` command; it is documented in §15.6.2 of the Maude 3.2.1 manual, available at maude-manual. (2) recall from pg. 11 of Lecture 20 that, as explained also in the Maude 3.2.1 manual, to perform narrowing-based model checking all rules in the module must be declared with the `[narrowing]` attribute.

2. Prove that `R&W` satisfies the *deadlock freedom* invariant.

   Hint. You may find it useful to understand **Method 1** in Appendix 3 to Lecture 20.