CS 476 Homework #11 Due 10:45am on 11/7

Note: Answers to the exercises listed below and all Maude code as well as screenshots of tool interactions should be emailed to `clarage2@illinois.edu`.

1. Solve **Ex**.21.1 in pg. 11 of Lecture 21. To avoid any ambiguities, the Lifting Lemma modulo B (which you do not need to prove but can *assume* when solving **Ex**.21.1) is explicitly stated below:

   Theorem (Lifting Lemma modulo B). Let (Σ, B, R) be a rewrite theory, $t \in T_\Sigma(X)$, and θ an R/B-*irreducible* substitution (i.e., if $x \in dom(\theta)$, then $\theta(x)$ cannot be rewritten with R modulo B). Then, for each rewrite step modulo B, $t\theta \rightarrow_{R/B} u$ there is a narrowing step modulo B, $t \overset{\alpha}{\rightsquigarrow}_{R/B} v$ and an R/B-irreducible substitution δ such that $v\delta = u$.

   Note that the above theorem extends in a straightforward manner to narrowing sequences modulo B,

   $$t \overset{\theta_1}{\rightsquigarrow}_{R/B} t_1 \ldots t_n \overset{\theta_{n+1}}{\rightsquigarrow}_{R/B} t_{n+1}$$

   which do indeed cover *all* R/B-rewriting computations $t\theta \rightarrow^*_{R/B} w$ as *instances.*

2. Recall the Readers and Writers mutual exclusion protocol in Lecture 19:

```
mod R&W is
  protecting NAT .
  sort Config .
  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
```

   Prove by narrowing-based symbolic model checking the following two invariants from the initial state `< 0, 0 >` which were only proved up to a 10^6 depth bound by explicit-state model checking in Lecture 19:

   - **Mutual exclusion**: readers and writers never access the resource simultaneously: only readers or only writers can do so at any given time.
   - **One writer**: at most one writer will be able to access the resource at any given time.

   Warning: Please, do not to fall into the pitfall of not giving the `[narrowing]` attribute to each of the rules in your module before giving `fvu-narrow` commands (see how this is done for the `BAKERY` protocol in pg. 14 of Lecture 21). If you forget to declare the `[narrowing]` attribute for each rule, what will happen is that *nothing will happen*, i.e., that no narrowing search will happen at all. Therefore you will get spurious `No solution` answers that do not mean anything and prove *nothing.*

   Extra Credit. You can earn up to 50% extra credit for this problem if, using some method among those described in Appendix 3 of Lecture 21, you can also prove the following additional invariant:

   - **Deadlock freedom**: there are no deadlocks.