Algebraic Specification in Maude

Rigorous Software Development
EUROPEAN MASTER IN SOFTWARE ENGINEERING/ MASTER IN SOFTWARE AND SYSTEMS
Universidad Politécnica de Madrid/IMDEA Software

December 9, 2015

To be turned in by January 15, 2016. Send them to me by mail to `jmarino@fi.upm.es`.

In order to deal with the following exercises you are assumed to have read chapters 1 and 2 of the Maude primer.

Exercise 1. Peano naturals – i.e. represented by means of `s` and `0` – are defined in the module `NAT` of the standard prelude. Define your own `MY-NAT` functional module defining operations `half`, `even` and `odd` reusing the sorts declared in `NAT` but not the arithmetic operations defined there.

Exercise 2. Add operations `quot` and `remd` for computing the quotient and remainder of the division of two natural numbers.

Exercise 3. (TRICKY) Your implementation of the `even` and `half` operations in Exercise 1 are most likely linear, i.e. the number of rewrites when reducing a term of the form `even(`N`)` is proportional to N. Define a functional module with an alternate version of Peano naturals again with constructors `s` and `0`, but now with an operation `even` that yields its result in a single rewrite step.

Exercise 4. A simplified phonebook allows only one telephone number per person. Define a module `PHONEBOOK` with operations for creating an empty phonebook (`create`), looking up a given person's number (`lookup`), deleting somebody's number (`delete`) and adding or updating somebody's number (`update`).

Exercise 5. Program an operation to compute Fibonacci numbers with reasonable efficiency in two different ways. Compare their running times and rewrites in computing $fib(1000)$.

Exercise 6. Define a module for binary search trees of naturals, where nodes in the left (resp. right) subtree are smaller (resp. greater) than the number at the root. Use these trees for defining an operation to sort lists of naturals.