# CS 476 Homework #7 Due 10:45am on 10/17

**Note:** Answers to the exercises listed below should be handed to the instructor *in hardcopy* and in *typewritten form* (latex formatting preferred) by the deadline mentioned above. You should also email the Maude code for Problem 2 to `hildenb2@illinois.edu`.

1. Solve Exercise 100 in *STACS*.

2. This problem is yet another good example of the motto:

   *Declarative Programming = Mathematical Modeling*

   Specifically, of how you can define an executable mathematical model in equational logic of the set $\mathcal{P}_{fin}(\mathbf{N})$ of *finite* sets of natural numbers with the membership $\_ \in \_$, containment $\_ \subseteq \_$ and set equality $\_ = \_$ predicates, and the operations of set difference, set intersection, and cardinality, all with the *exact same meaning* given to these notions in the first few chapters of *STACS*.

   Since your *mathematical model* should be specified by an equational theory,

   > *No use should be made of the* `owise` *attribute in equations. Likewise, no use should be made of the built-in equality predicate* `==` *in any equations.*

   Finite sets of natural numbers are built by means of a binary associative and commutative set union constructor $\_,\_$ which has the empty set `mt` as its identity element. Then you are asked to write equations defining the following functions:

   (a) set union
   (b) an equality predicate on numbers
   (c) the containment predicate $\_ \subseteq \_$ on sets
   (d) the membership relation $\_ \in \_$ of a number in a set
   (e) an equality predicate between sets
   (f) set difference between two sets
   (g) intersection of sets
   (h) cardinality of a sets.

   **Hints**:

   - The built-in module `NAT` is included for your convenience because: (i) it supports decimal notation and also Peano notation: 3 can be written both as `3` and as `s(s(s(0)))`, which is very convenient: you can for example define the equality predicate between naturals just using the Peano notation; (ii) it imports the `BOOL` module, so you have at your disposal all the Boolean operations, which can be useful when defining some of the predicates; and (iii) `BOOL` itself imports the if-then-else-fi operator, which again can be helpful when defining some functions.
   - The order in which the functions are introduced gives you a hint that some functions earlier in the list may be useful as auxiliary functions for other functions later down the list.

- Programming modulo axioms of associativity, commutativity, and identity is very powerful and allows writing very short programs. For example, the eight functions in this example can be defined with just 18 equations. However, with this power comes also the danger of non-termination: unless you think carefully about ways in which your equations could loop when one of the variables of sort `Set` in a lefthand side is instantiated to the empty set, you may write non-terminating equations.
- You should *never* rely on the order in which equations may be applied. If you do, this will be a clear sign that your solution is *wrong*, since you will *not* have defined a *correct* mathematical model of finite sets of natural numbers.
- Of course, the `MULTISET-ALGEBRA` example in Homework#4 should be a good source of ideas. However, the algebra of sets differs in some subtle ways from that of multisets, so you cannot just "cut and paste" willy-nilly form Homework#4: you should think carefully about what is the *correct definition* of each operation in the case of sets.

```
fmod SET-ALGEBRA is
  protecting NAT .
  sort Set .
  subsort Nat < Set .
  op mt : -> Set [ctor] .                              *** empty set
  op _,_ : Set Set -> Set [ctor assoc comm id: mt] . *** set union
  op _~_ : Nat Nat -> Bool [comm] .       *** equality predicate on naturals
  op _C=_ : Set Set -> Bool .             *** set containment
  op _in_ : Nat Set -> Bool .             *** set membership
  op _~_  : Set Set -> Bool [comm] .      *** equality predicate on sets
  op _\_  : Set Set -> Set .              *** set difference
  op _/\_ : Set Set -> Set .              *** set intersection
  op |_| : Set -> Nat .                   *** set cardinality

  vars N M : Nat .  vars U V W : Set .

*** add your equations here

endfm

red 5 ~ 12 .                  *** should be false
red 15 ~ 15 .                 *** should be true

red (3,3,4,4,4,2,2,9) \ (3,3,3,4,2,7) .  *** should be 9

red (3,3,4,4,4,2,2,9) C= (3,3,3,4,2,7) .  *** should be false

red (3,3,4,4,2,2,9) C= (3,4,2,7,9) .  *** should be true

red 3 in (3,3,4,4,7) .                      *** should be true

red 9 in (3,3,4,4,7) .                      *** should be false

red (3,3,4,4,4,2,2,7) ~ (3,3,3,4,2,7) .  *** should be true

red (3,3,3,4,4,4,2,2,7,9) /\ (3,3,3,3,4,4,2,7,7,10) . *** should be 2,3,4,7

red | 3,3,4,4,4,2,2,9 | .       *** should be 4
```

You can retrieve this module as a "skeleton" on which to give your answer from the course web page. Also, send a file with your module to `hildenb2@illinois.edu`.