CS 476 Homework #4 Due 10:45am on 9/22

Note: Answers to the exercises listed below (in typewritten form, preferably using Latex) as well as the Maude code for Problem 2, should be emailed by the above deadline to nishant2@illinois.edu.

1. The addition function on the natural numbers:

\[ + : \mathbb{N}^2 \rightarrow \mathbb{N} \]

is a relation, and therefore a subset \( +_\mathbb{N} \subset \mathbb{N}^2 \times \mathbb{N} \). In decimal notation this subset cannot be explicitly described without invoking the decimal addition algorithm to specify the set defined by this function. However, a nice feature of the Peano representation of the naturals is that \( +_\mathbb{N} \) can be explicitly described as a set. It is the set:

\[
+_\mathbb{N} = \{((n, 0), n) \in \mathbb{N}^2 \times \mathbb{N} \mid n \in \mathbb{N}\} \cup \{(n, s(m)), s^m(n) \in \mathbb{N}^2 \times \mathbb{N} \mid n, m \in \mathbb{N}\}.
\]

where, by definition, \( s^0(n) = s(n) \), and \( s^m(n) = s(s^{m-1}(n)) \), for \( m > s(0) \).

Consider now the following Maude functional module (in prefix notation):

```maude
fmod NATURAL is
  sort Nat.
  op 0 : -> Nat [ctor] .
  op s : Nat -> Nat [ctor] .
  op + : Nat Nat -> Nat .
  vars N M : Nat .
  eq +(N,0) = N .
  eq +(N,s(M)) = s(+(N,M)) .
endfm
```

Adopting the Peano notation, any natural number \( n \in \mathbb{N} \) is exactly a constructor term in the above module, i.e., \( n \) is either 0, or has the form \( s^k(0) \) for some \( k \geq 1 \).

You are asked to do two things:

(A). Prove the following theorem:

**Theorem.** For any \( n, m \in \mathbb{N} \) in Peano notation, the term \( +(n, m) \) has a unique terminating sequence of equality steps:

\[ +(n, m) = t_1 = t_2 = \ldots = u \]

such that each step in the sequence is obtained by applying one of the two equations in \( \textsc{natural} \) from left to right as a simplification rule.\(^1\) Furthermore, the term \( u \) in which the sequence terminates is precisely the constructor term \( +_\mathbb{N}(n, m) \in \mathbb{N} \), where \( +_\mathbb{N} \) is the addition function on natural numbers in Peano notation specified set-theoretically above as a binary relation \( +_\mathbb{N} \subset \mathbb{N}^2 \times \mathbb{N} \).

**Hint.** Use induction!

(B). Use the above theorem [plus the (correct) assumption that \( \textsc{natural} \) satisfies all the executability conditions needed for it to define a canonical term algebra] to show that for \( (\Sigma, E) \) the equational theory specified by the above module \( \textsc{natural} \), its canonical term algebra \( \mathbb{C}_{\Sigma/E} \) is exactly what one would expect: the algebra of the natural numbers \( \mathbb{N} \) in Peano notation, with the standard interpretation for the symbols \( \{0, s, +\} \) in \( \mathbb{N} \).

\(^1\)What this means is intuitively obvious: we have seen various examples. But, in any case, this process of left-to-right simplification has been formally defined as term rewriting with the rules \( +(N,0) \rightarrow N \) and \( +(N,s(M)) \rightarrow s(+(N,M)) \) in Lecture 5.
2. This exercise is about using lists of naturals to represent sets of naturals [in an obviously non-unique way, since lists can have repeated elements, and, even if they do not, list elements may appear in different orders]. Specifically, you are asked to define functions:

- \textbf{insert} to insert a number into a set.
- a predicate \texttt{in} to test whether a number belongs to a set.
- \texttt{U} to compute the union of two sets.
- \texttt{simplify} to obtain a list representation of a set that has no repeated elements.
- \texttt{\textbackslash{}\textbackslash{}} to compute the intersection of two sets.
- \texttt{-} to compute the difference of two sets.
- \texttt{equal-sets} to test whether or not two lists represent the same set.

You can do so by adding the needed equations defining such functions [plus those for any other auxiliary functions that you may need] to the module below, which imports \texttt{NAT}, the built-in naturals. This ensures that you have various functions, such as \texttt{if\_then\_else\_fi}, order comparison between numbers, Boolean operations, and the built-in equality predicate \texttt{==} (for both numbers and lists) already available to you.

```
fo mod LIST-REPRESENTATION-OF-SETS is
    protecting \texttt{NAT} .
    sort \texttt{List} .
    op \texttt{nil} : \to \texttt{List} [ctor] .
    op \texttt{\_;\_} : \texttt{Nat List} \to \texttt{List} [ctor] .

    \vars N M : \texttt{Nat} . \vars L L1 L2 : \texttt{List} .
    op \texttt{insert} : \texttt{Nat List} \to \texttt{List} . \ *** inserts a number into a list viewed as a "set"
        *** add your equations here
    op \texttt{in} : \texttt{Nat List} \to \texttt{Bool} . \ *** checks if a number is in the list viewed as a "set"
        *** add your equations here
    op \texttt{U} : \texttt{List List} \to \texttt{List} . \ *** computes a list representing "set" union
        *** add your equations here
    op \texttt{simplify} : \texttt{List} \to \texttt{List} . \ *** returns a list representing the set with no repetitions
        *** add your equations here
    op \texttt{\textbackslash{}\textbackslash{}} : \texttt{List List} \to \texttt{List} . \ *** computes a list representing "set" intersection
        *** add your equations here
    op \texttt{-} : \texttt{List List} \to \texttt{List} . \ *** computes a list representing "set" difference
        *** add your equations here
    op \texttt{equal-sets} : \texttt{List List} \to \texttt{Bool} . \ *** checks if two lists represent the same set
        *** add your equations here
endfm
```
Since many different lists can represent the same set, there are of course different ways of defining the above functions. For example, both 2; 3; 1; nil and 2; 3; 1; nil are correct representations of the set union of the “sets” 2; 3; nil and 3; 1; nil. You just need to make sure that your definition of each function corresponds to one of the possible ways of correctly representing that set-theoretic function.

You can retrieve from the course web page this module as a “skeleton” on which to give your answer. Also, send a file with your module and tests cases to nishant2@illinois.edu.