CS 476 Homework #4 Due 10:45am on 10/21

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 to (`skeirik2@illinois.edu`) the Maude code for the exercises requiring that.

1. Solve **Ex.** 10.4 in Lecture 10.

2. The equational theory of commutative rings is one of the most common in mathematics. It axiomatizes the operations of addition, minus, and multiplication. The integers, the rationals, the reals and the complex numbers are all commutative rings.

   Given an equational theory (Σ, E), Maude can specify that theory itself (describing the class of all (Σ, E)-algebras) as a *functional theory* with the keywords: `fth` (Σ, E) `endfth`. For example, for (Σ, E) the theory of commutative rings then `fth` (Σ, E) `endfth` specifies the class of *all* commutative rings, including the integers, the integers modulo k, the rationals, the reals, the complex numbers, etc.

   Here is the theory of commutative rings:

```
fth COMM-RING is
  sort Ring .
  op _+_ : Ring Ring -> Ring [assoc comm] .
  op _*_ : Ring Ring -> Ring [assoc comm] .
  ops 0 1 : -> Ring .
  op - : Ring -> Ring .
  vars x y z : Ring .
  eq x + 0 = x .
  eq 1 * x = x .
  eq x + -(x) = 0 .
  eq x * (y + z) = (x * y) + (x * z) .
endfth
```

   The last equation is called the *distributivity* equation, relating multiplication and addition.

   In general, given an equational theory (Σ, E) we do *not* have a *decision procedure* that, given an equation $u = v$, can answer the question of whether or not $(\Sigma, E) \vdash u = v$, that is, whether or not $u = v$ is a *theorem* of the theory (Σ, E). That is, in general the problem of whether $(\Sigma, E) \vdash u = v$ is only *semi-decidable*: if indeed $(\Sigma, E) \vdash u = v$ we can find out by enumerating all proofs of $u = w$ for any w, because then a proof of $u = v$ will eventually show up in such an enumeration. However, if $u = v$ is *not* a theorem of (Σ, E) we may search forever for such a proof, never finding it.

   However, if we can manage to decompose E as $E = E_0 \cup B$ with $\vec{E_0}$ confluent and terminating modulo B, for B axioms like associativity, commutativity and identity, we *do* get a decision procedure for $(\Sigma, E) \vdash u = v$, namely, by reducing u and v to canonical form and testing whether or not they are equal modulo B. If they are, then $u = v$ is a theorem of (Σ, E), if they are not, then $u = v$ is *not* a theorem of (Σ, E). This is all thanks to the Church-Rosser Theorem (which you just proved as Problem 2 above), which gives us the equivalence:

$$(\Sigma, E) \vdash u = v \quad \Leftrightarrow \quad u \downarrow_{\vec{E_0}/B} v$$

   This raises the following question: is theoremhood in the theory of commutative rings *decidable*? It will be if we can make the equations of the theory of commutative rings confluent and terminating modulo AC. The

above equations are terminating but not confluent. However, they *can* be made confluent (and still remain terminating) by adding a few extra equations that are themselves also theorems of the theory of commutative rings.

The following theory is *almost* confluent (it is indeed terminating). It is just missing one equation. It is enclosed in parentheses so that you can enter it in the Church-Rosser Checker and see what critical pairs it has. You are asked to add the missing equation and to check that it is now confluent. Yo do not need to check that it is terminating, but it will indeed be terminating as well if you add the right equation.

```
(fth COMM-RING is
  sort Ring .
  op _+_ : Ring Ring -> Ring [assoc comm] .
  op _*_ : Ring Ring -> Ring [assoc comm] .
  ops 0 1 : -> Ring .
  op - : Ring -> Ring .
  vars x y z : Ring .
  eq x + 0 = x .
  eq x * 1 = x .
  eq 1 * x = x .
  eq x + -(x) = 0 .
  eq -(0) = 0 .
  eq -(-(x)) = x .
  eq -(x + y) = -(x) + -(y) .
  eq x + (y + -(x)) = y .
  eq x * (y + z) = (x * y) + (x * z) .
  eq x * -(y) = -(x * y) .
endfth)
```

Include in your answer the competed specification and a screenshot of the successful reply from the Church-Rosser Checker.

Of course, for any equational theory (Σ, E) we can write in Maude two specifications:

- `fth` (Σ, E) `endfth`, denoting the class of *all* (Σ, E)-algebras, and
- `fmod` (Σ, E) `endfm`, denoting the *initial* (Σ, E)-algebra.

Since after you add the missing equation all the executability conditions of a functional module will be satisfied, you can define in Maude the functional module:

```
fmod COMM-RING-INITIAL is
  sort Ring .
  op _+_ : Ring Ring -> Ring [assoc comm] .
  op _*_ : Ring Ring -> Ring [assoc comm] .
  ops 0 1 : -> Ring .
  op - : Ring -> Ring .
  vars x y z : Ring .
  eq x + 0 = x .
  eq x * 1 = x .
  eq 1 * x = x .
  eq x + -(x) = 0 .
  eq -(0) = 0 .
  eq -(-(x)) = x .
  eq -(x + y) = -(x) + -(y) .
  eq x + (y + -(x)) = y .
  eq x * (y + z) = (x * y) + (x * z) .
  eq x * -(y) = -(x * y) .
```

```
  *** add your missing equation here to make it confluent
endfm
```

which specifies the initial algebra of the theory of commutative rings.

Please play with this module a little to experiment with it, and answer the following question:

> *Which well-know ring is the initial algebra of the theory of rings? That is, which well-known ring is defined by the functional module* `COMM-RING-INITIAL` *when you add the missing equation?*

3. Consider the following module, which was already used in Homework 3, of lists with an append constructor function that is associative and has an identity, but where associativity and identity are explicitly defined by equations:

```
(fmod LIST-EXAMPLE is
 sorts Element List .
 subsorts Element < List .
 op a : -> Element [ctor] .
 op b : -> Element [ctor] .
 op c : -> Element [ctor] .
 op nil : -> List [ctor] .
 op _;_  : List List -> List .
 op _;_  : Element List -> List [ctor] .
 vars L P Q : List .
 eq (L ; P) ; Q = L ; (P ; Q) .
 eq L ; nil = L .
 eq nil ; L = L .
endfm)
```

In Homework 3 you were asked to use the Maude Church-Rosser Checker to establish the local confluence of this module (which with termination implies confluence). But can you do it yourself? That is: you are asked to:

- compute all the critical pairs of this module, give a complete list of them, and explain what computations you have performed to arrive at that list: you can do it by hand or can use Maude to help you with computing order-sorted unifiers by calling the `unify` command (explained in Section 12.4 of the Maude 2.6 Manual available in the Maude web page).
- show that each critical pair can be joined; you can again do it by hand or can use Maude to automate the check by using the `_==_` operator to compute that for each critical pair (t, t') both terms hace the same canonical form.

4. Consider the module

```
fmod LIST-of-NAT is protecting NAT .
  sort List .
  subsort Nat < List .
  op nil : -> List [ctor] .
  op _;_ : List List -> List [ctor assoc] .
  op length : List -> Nat .
  var N : Nat .
  var L : List .
  eq nil ; L = L .
  eq L ; nil = L .
  eq length(nil) = 0 .
  eq length(N) = 1 .
  eq length(N ; L) = 1 + length(L) .
endfm
```

use the ITP to prove the following theorem:

```
(goal listlength : LIST-of-NAT
  |- A{L:List ; L':List}
     ((length(L ; L')) = (length(L) + length(L'))) .)
```

Note: You can retrieve both the module and the goal from the course web page.