CS 476 Homework #12 Due 10:45am on 4/20

Note: Answers to the exercises listed below and screenshots of all tool interaction should be sent by email to `reedoei2@illinois.edu`.

1. The most basic formulas of equational logic are of course equations $u = v$. The *quantifier-free* (QF) *formulas* are Boolean combinations of such equations by negation ($\neg$) (where $\neg(u = v)$ is abbreviated to $u \neq v$), conjunctions ($\wedge$) and disjunction ($\vee$). For example, $u \neq u' \vee (v = v' \wedge w = w')$ is a QF formula, which given the Boolean equivalence $\neg(A) \vee B \equiv A \Rightarrow B$, could also be written as $u = u' \Rightarrow (v = v' \wedge w = w')$. The notion of satisfaction $\mathbb{A} \models u = v$ of an equation by an algebra, extends naturally to QF formulas, say, φ, ψ, etc., as follows: $\mathbb{A} \models \varphi$ iff for all assignments $a \in [X \to A]$, $(\mathbb{A}, a) \models \varphi$ holds, where: (i) if $\varphi \equiv u = v$ this exactly means $u\,a = v\,a$, i.e., the usual notion of equation satisfaction. Otherwise, satisfaction is defined inductively by:

   - $(\mathbb{A}, a) \models \neg\varphi$ iff $(\mathbb{A}, a) \models \varphi$ does *not* hold.
   - $(\mathbb{A}, a) \models \varphi \wedge \psi$ iff $(\mathbb{A}, a) \models \varphi$ and $(\mathbb{A}, a) \models \psi$.
   - $(\mathbb{A}, a) \models \varphi \vee \psi$ iff $(\mathbb{A}, a) \models \varphi$ or $(\mathbb{A}, a) \models \psi$.

   More generally, we can consider arbitrary formulas in *first-order logic*, where we do not only close the basic formulas (i.e., equations) by Boolean combinations, but also by universal ($\forall x$) and existential ($\exists x$) quantification. We can likewise define the notion of satisfaction $\mathbb{A} \models \varphi$ for φ any first-order formula, but we shall not do so here. There is, however, a very easy and very useful thing to say at this point. For φ a QF formula, let $\forall\varphi$ denote its *universal closure*, i.e., we universally quantify *all* the variables appearing in φ (for example, the goals given to the ITP are always expressed as universal closures). Then we have the equivalence:

$$\mathbb{A} \models \varphi \quad \Leftrightarrow \quad \mathbb{A} \models \forall\varphi$$

   which reduces satisfaction of universal closures of QF formulas to that of QF formulas themselves.

   If you have understood the above definition of satisfaction of a QF formula in an algebra $\mathbb{A}$, you can surely do the following exercise: Consider the unsorted signature $\Sigma_D = \{0, s\}$ called the "Dedekind" signature in STACS. Now consider the following very simple functional module:

```
fmod MOD-2 is
 sort Natural .
 op 0 : -> Natural [ctor] .
 op s : Natural -> Natural [ctor] .
 var x : Natural .
 eq s(s(x)) = x .
endfm
```

   Since the equation `s(s(x)) = x` is term-size-decreasing, this module is terminating. It is also very easy to check that it is confluent: you can check the critical pairs themselves (note that there is a non-trivial case of "self-overlap" of the rule with itself at position 1); or you can do so automatically with Maude's the Church-Rosser Checker. As you may have guessed this is one of the simplest possible models of the natural numbers modulo 2. Furthermore, it does have not only an initial algebra $\mathbb{T}_{\Sigma_D/E_2}$, where E_2 is the single equation `s(s(x)) = x`, but, thanks to being convergent, it has also a canonical term algebra $\mathbb{C}_{\Sigma_D/E_2}$.

   As a simple exercise to test your understanding of the notion of satisfaction of a QF formula, you are asked to prove two simple theorems about the initial algebra $\mathbb{T}_{\Sigma_D/E_2}$. Prove (by yourself, not using any tools) that:

(a) $\mathbb{T}_{\Sigma_D/E_2} \models s(x) \neq x$

(b) $\mathbb{T}_{\Sigma_D/E_2} \models x = 0 \vee x = s(0)$.

Hints. (A) Since we have the isomorphism $\mathbb{T}_{\Sigma_D/E_2} \cong \mathbb{C}_{\Sigma_D/E_2}$ and isomorphic algebras satisfy the same formulas, in proving (a) and (b) above you can replace $\mathbb{T}_{\Sigma_D/E_2}$ by $\mathbb{C}_{\Sigma_D/E_2}$. (B) Note that: (B.1) all notions related to formula satisfaction in an algebra $\mathbb{A}$ are expressed in terms of assignments $a \in [X \to A]$, (B.2) note that the cardinality of the algebra $\mathbb{C}_{\Sigma_D/E_2}$ is exactly 2, and (B.3) note that, even if X is an infinite set of variables, the only variables *that matter* for a given formula φ are those in the finite set Y of variables appearing in φ: that is, for any two assignments $a, a' \in [X \to A]$ for an algebra $\mathbb{A}$ such that $a|_Y = a'|_Y$ we have the equivalence: $(A, a) \models \varphi$ iff $(\mathbb{A}, a') \models \varphi$.

2. In Inductive Theorem Proving, one would like to *use* theorems one has already proved to prove new theorems. This can of course be done by using such previous theorems as *lemmas*. But often one can do even better than that: one can, so to speak, *internalize* an already proved lemma *into the functional module* used to prove the next theorem. This is because of the following, important result, whose proof will be provided elsewhere:

   Lemma. For any equational theory (Σ, E) with Σ a sensible, kind complete signature with no empty sorts, if $\mathbb{T}_{\Sigma/E} \models u = v$, then $\mathbb{T}_{\Sigma/E} = \mathbb{T}_{\Sigma/E \cup \{u=v\}}$.

   The way the above Lemma can be used to "internalize" a proved inductive theorem $u = v$ is that if our original functional module was, say, `fmod` (Σ, E) `endfm` and we now want to prove another inductive theorem, say $t = t'$, we can do so starting with the module: `fmod` $(\Sigma, E \cup \{u = v\})$ `endfm` where the proved theorem $u = v$ has been "internalized." Of course, this might break the executability conditions of our original module, so this is not always wise to do. But we can safely do this when:[1]

   - $u = v$ is an *attribute* like associativity, or commutativity, or identity, which we could add declaring such new attribute in the module (assuming the module's equations remain terminating), or
   - the set of equations $E \cup \{u = v\}$ is still (ground) confluent and terminating.

   For example, in Lecture 13 we proved *associativity and commutativity* of natural number addition. Why not add them as *axioms* to then prove *distributivity of multiplication*? That is, in the following module, where such axioms have been added and multiplication is defined:

```
fmod NATURAL is
  sort Natural .
  op 0 : -> Natural [ctor] .
  op s : Natural -> Natural [ctor] .
  op _+_ : Natural Natural -> Natural [assoc comm] .
  op _*_ : Natural Natural -> Natural .
  vars N M : Natural .
  eq N + 0 = N .
  eq N + s(M) = s(N + M) .
  eq N * 0 = 0 .
  eq N * s(M) = N + (N * M) .
endfm
```

   use the Maude ITP to prove the distributivity property goal:

```
(goal dist : NATURAL
|- A{N:Natural ; J:Natural ; K:Natural}
((N * (J + K)) = ((N * J) + (N * K))) .)
```

   Note. Detailed instructions to use the ITP can be found in the Course's web page. Here is an additional remark to help you use the tool. You may get some warnings when running `loop init-itp` like, for example,

[1] Much weaker conditions can be given for internalizing $u = v$, but those given here will suffice for our present purposes.

```
Warning: sort declarations for operator _;_ failed preregularity check on 15 out of 2116 sort tuples. First such
Warning: sort declarations for operator if_then_else_fi failed preregularity check on 12 out of 4232 sort tuples.
Warning: sort declarations for associative operator _;_ are non-associative on 153 out of 97336 sort triples. Firs
```

You can safely ignore these warnings.

For Extra Credit. (You can double your grade for Problem 2 if you solve this additional exercise). But why not doing the same again? Suppose that we now want to prove that multiplication is *associative*. If you try to do it by hand (or with the ITP), you will soon see that you need to use distributivity. But we have already *proved* distributivity! Why not add it as a rule to the above module to get the extended module:

```
fmod NATURAL+DIST is
  sort Natural .
  op 0 : -> Natural [ctor] .
  op s : Natural -> Natural [ctor] .
  op _+_ : Natural Natural -> Natural [assoc comm] .
  op _*_ : Natural Natural -> Natural .
  vars N M J K : Natural .
  eq N + 0 = N .
  eq N + s(M) = s(N + M) .
  eq N * 0 = 0 .
  eq N * s(M) = N + (N * M) .
  eq N * (J + K) = (N * J) + (N * K) .  *** dist added as proved lemma
endfm
```

Use now the Maude ITP to prove in this extended module the associativity of multiplication goal:

```
(goal *-assoc : NATURAL+DIST
|- A{N:Natural ; J:Natural ; K:Natural}
(((N * J) * K) = (N * (J * K))) .)
```

You can just extract the two modules and their corresponding goals from the Latex, but the files for these will also be posted with this homework. You should include a screenshot of your interactions with the ITP and email your code as well to `reedoei2@illinois.edu`.