Assignment 4 – COMP 426: Automated Reasoning

Fall 2007
Due Oct 12th 2007

Use Tutch to check your implementation of the following problems.
A sample solution looks like this:

```
% S. Double a Natural Number
%
%  Give a specification and implementation for the function double,
%  which doubles a natural number.
%
%val double : nat -> nat
% Solution:
% a) specification:
%
%  double : nat -> nat
%  double 0 = 0
%  double sx = ss(double x)
%
% b) implementation:

val double : nat -> nat =
 fn x => rec x
  of f 0 => 0
  | f(s x’) => s (s (f x’))
  end;
```

Check the file `ass04.tut` on the course website.

Exercise 1: Boolean Functions **(20pt)**

Give specifications and term implementations for the following boolean operations: iff, xor (exclusive or), and nand.

Exercise 2: Maximum Function **(30pts)**

Implement the maximum function `max : nat -> nat -> nat` that returns the maximum of two natural numbers. Implement first the auxiliary function which tests whether x is greater or equal to y.

Exercise 3: Data-type `exp` for arithmetic expressions **(50pts)**
Note, that you cannot check your answers with Tutch.

1. Define an inductive data type `exp` of arithmetical expressions over natural numbers, the terms of which are expressions like $2 * (3 + 1)$, $(4 + 22) * (3 + 9)$, etc.

   Give formation, introduction, and elimination rules involving two constructors `Pxy` and `Txy` (for plus and times). Define also a base constructor `numb` which converts a natural number into an expression.

2. Using the previous data-type definition, give specification and implementations for each of the following functions.

   – Define the function `count:exp -> nat`. The function `count(e)` counts the number of numerals occurring in the expresion `e`.

   – Define the function `eval:exp -> nat`. The function `eval(e)` evaluates the expression `e` and returns a natural number as the final result.