Semantics, WS 2011-2012: Solution for Assignment 1

Prof. Dr. Gert Smolka, Dr. Chad Brown

Read the web pages for the course. Install the Coq system on your computer. Read Sections 2.1-2.4 of the lecture notes for "Computational Logic".Then do the following exercises. Also read through the chapter "Basics" of the "Software Foundations" text.

Exercise 1.1 (Disjunction) A boolean disjunction $x \vee y$ yields *false* if and only if both x and y are *false*.

a) Define disjunction as a function $orb : bool \to bool \to bool$ in Coq.

b) Prove the de Morgan law $\neg(x \vee y) = \neg x \wedge \neg y$ in Coq.

Solution to Exercise 1.1 (Disjunction)

```
Definition orb (x y : bool) : bool :=
match x with
| false => y
| true => true
end.

Lemma ex_2_2_2b (x y : bool) : negb (orb x y) = andb (negb x) (negb y).
Proof.
destruct x ; destruct y ; reflexivity.
Qed.
```

Exercise 1.2 Define functions as follows.

a) A function $power : nat \to nat \to nat$ that yields x^n for x and n.

b) A function $fac : nat \to nat$ that yields $n!$ for n.

c) A function $even : nat \to bool$ that tests whether its argument is even.

d) A function $mod3 : nat \to nat$ that yields the remainder of x on division by 3.

Solution to Exercise 1.2

```
Fixpoint power (x n : nat) : nat :=
match n with
| O => (S O)
| S n' => mul x (power x n')
end.
```

```
Fixpoint fac (n : nat) : nat :=
match n with
| O => (S O)
| S n' => mul n (fac n')
end.

Fixpoint even (n : nat) : bool :=
match n with
| O => true
| S n => negb (even n)
end.

(*** Alternative Solution: ***)
(***
Fixpoint even (n : nat) : bool :=
match n with
| O => true
| S O => false
| S (S n') => even n'
end.
***)

Fixpoint mod3 (n : nat) : nat :=
match n with
| S (S (S n)) => mod3 n
| n => n
end.

(*** Alternative Solution: ***)
(***
Fixpoint mod3 (n : nat) : nat :=
match n with
| O => O
| S O => S O
| S (S O) => S (S O)
| S (S (S n')) => mod3 n'
end.
***)
```

Exercise 1.3 Prove the following lemmas.

```
Lemma mul_O (x : nat) : mul x O = O.

Lemma mul_S (x y : nat) : mul x (S y) = add (mul x y) x.

Lemma mul_com (x y : nat) : mul x y = mul y x.
```

Lemma mul_dist (x y z: nat) : mul (add x y) z = add (mul x z) (mul y z).

Lemma mul_asso (x y z: nat) : mul (mul x y) z = mul x (mul y z).

Solution to Exercise 1.3

```
Lemma mul_O (x:nat): mul x O = O.
Proof.
induction x.
reflexivity.
simpl. rewrite IHx. reflexivity.
Qed.

Lemma mul_S (x y:nat): mul x (S y) = add (mul x y) x.

(***
(*** Solution 1 ***)
Proof.
induction x; simpl. reflexivity.
rewrite IHx. rewrite add_S. rewrite add_asso.
rewrite add_S. rewrite add_asso. rewrite (add_com y).
reflexivity.
Qed.
***)

(*** Solution 2 ***)
Proof.
induction x; simpl. reflexivity.
rewrite IHx.
rewrite add_asso. rewrite add_asso. f_equal.
rewrite add_S. rewrite add_S. rewrite add_com. reflexivity.
Qed.

Lemma mul_com (x y:nat): mul x y = mul y x.
Proof.
induction x ; simpl.
rewrite mul_O. reflexivity.
rewrite mul_S. rewrite IHx. reflexivity.
Qed.

Lemma mul_dist (x y z:nat): mul (add x y) z = add (mul x z) (mul y z).

(***
(*** Solution 1 ***)
Proof.
induction x ; simpl. reflexivity.
```

```
rewrite IHx. rewrite add_asso.  rewrite add_asso.
rewrite (add_com (mul y z)). reflexivity.
Qed.
***)

(*** Solution 2 ***)
Proof.
induction x ; simpl. reflexivity.
rewrite IHx.  rewrite add_asso.  rewrite add_asso.
f_equal. rewrite add_com. reflexivity.
Qed.

Lemma mul_asso (x y z:nat): mul (mul x y) z = mul x (mul y z).
induction x ; simpl. reflexivity.
rewrite mul_dist. rewrite IHx. reflexivity.
Qed.
```

Exercise 1.4 Consider the following Coq proof. After each tactic in the proof, there will be a number of goals. After each tactic give the number of goals and for each of these goals give the assumptions of the goal and the claim of the goal.

```
Lemma add_S' (x y : nat) :  add x (S y) = S (add x y).

Proof.
induction x.
simpl.
reflexivity.
simpl.
rewrite IHx.
reflexivity.
Qed.
```

Solution to Exercise 1.4

```
Proof.
```

There is one goal:

$$\frac{\begin{array}{c} x : nat \\ y : nat \end{array}}{add\, x\, (S\, y) \;=\; S\, (add\, x\, y)}$$

```
induction x.
```

There are two goals:

$$\frac{y : nat}{add\,O\,(S\,y) \;=\; S\,(add\,O\,y)}$$

and

$$\frac{\begin{array}{c}x : nat\\ y : nat\\ IHx : add\,x\,(S\,y) \;=\; S\,(add\,x\,y)\end{array}}{add\,(S\,x)\,(S\,y) \;=\; S\,(add\,(S\,x)\,y)}$$

simpl.

There are two goals:

$$\frac{y : nat}{S\,y \;=\; S\,y}$$

and

$$\frac{\begin{array}{c}x : nat\\ y : nat\\ IHx : add\,x\,(S\,y) \;=\; S\,(add\,x\,y)\end{array}}{add\,(S\,x)\,(S\,y) \;=\; S\,(add\,(S\,x)\,y)}$$

reflexivity.

There is one goal:

$$\frac{\begin{array}{c}x : nat\\ y : nat\\ IHx : add\,x\,(S\,y) \;=\; S\,(add\,x\,y)\end{array}}{add\,(S\,x)\,(S\,y) \;=\; S\,(add\,(S\,x)\,y)}$$

simpl.

There is one goal:

$$\frac{\begin{array}{c}x : nat\\ y : nat\\ IHx : add\,x\,(S\,y) \;=\; S\,(add\,x\,y)\end{array}}{S\,(add\,x\,(S\,y)) \;=\; S\,(S\,(add\,x\,y))}$$

rewrite IHx.

There is one goal:

$$\frac{\begin{array}{c} x : nat \\ y : nat \\ IH\,x : add\,x\,(S\,y) \;=\; S\,(add\,x\,y) \end{array}}{S\,(S\,(add\,x\,y)) \;=\; S\,(S\,(add\,x\,y))}$$

```
reflexivity.
```

There are no goals.