Construcción Formal de Programas en Teoría de Tipos

Primer Parcial - Octubre de 2015

Problema 1. Pruebe el siguiente lema usando lógica clásica:

```
Section Ej1.

Variables P R : Prop.

Lemma lema1_1 : (P -> R) \/ (R -> P).

End Ej1.
```

Problema 1. Considere el siguiente problema. Un zoológico tiene las siguientes reglas para sus animales:

Regla 1: Los animales cuadrúpedos no nadan.
Regla 2: Los animales herbívoros vuelan.
Regla 3: Los animales que no toman leche deben ser herbívoros.
Regla 4: Los animales vuelan o no toman leche.
Regla 5: Los animales que vuelan son herbívoros y cuadrúpedos.
Regla 6: Los animales nadan si y sólo si son herbívoros.

Demuestre que las reglas son contradictorias, o sea que ningún animal puede estar en ese zoológico, sin usar tácticas automáticas ni lógica clásica.

Problema 3. Sean T y U predicados binarios sobre elementos de un conjunto C. Pruebe en Coq sin usar tácticas automáticas, ni lógica clásica, el siguiente lema:

$$(\forall x\ y.(U(x, y) \rightarrow \sim T(x, y)) \ \wedge\ \exists z.T(z, z)) \ \rightarrow\ \exists w.\sim U(w, w)$$

Problema 4. Considere las siguientes declaraciones:

```
Section Ej4.

Variable Bool: Set.
Variable TRUE : Bool.
Variable FALSE : Bool.

Variable Not : Bool -> Bool.
Variable Or : Bool -> Bool -> Bool.
Variable And : Bool -> Bool -> Bool.

Axiom BoolVal : forall b : Bool, b = TRUE \/ b = FALSE.

Axiom NotTrue : Not TRUE = FALSE.
Axiom NotFalse : Not FALSE = TRUE.

Axiom AndTrue : forall b : Bool, And TRUE b = b.
Axiom AndFalse : forall b : Bool, And FALSE b = FALSE.

Axiom OrTrue : forall b : Bool, Or TRUE b = TRUE.
Axiom OrFalse : forall b : Bool, Or FALSE b = b.
```

Bajo el contexto previo, demuestre en Coq los siguientes lemas:

```
Lemma lema4_1 : forall b : Bool, Not (Not b) = b.

Lemma lema4_2 : forall b1 b2 : Bool, Not (Or b1 b2) = And (Not b1) (Not b2).

Lemma lema4_3 : forall b1 : Bool, exists b2 : Bool, Or b1 b2 = Or (Not b1) (Not b2).

End Ej4.
```

Problema 5. Considere la siguiente definición del constructor de tipos Array (paramétrico en el largo, y de elementos de un tipo genérico) y los siguientes operadores:

```
Parameter Array : Set -> nat -> Set.
Parameter emptyA : forall X : Set, Array X 0.
Parameter addA : forall (X : Set) (n : nat), X -> Array X n -> Array X (S n).
```

Considere asimismo el constructor de tipos `Matrix` de matrices *escalonadas* de *n* columnas (de elementos de un tipo genérico), donde la primera columna contiene un sólo elemento, la segunda columna 2 elementos, la tercera 3 y así sucesivamente, de tal manera que la columna *n* posee *n* elementos.

```
Parameter Matrix : Set -> nat -> Set.
```

a) Defina el tipo del operador `emptyM` que permita representar una matriz escalonada vacía (sin elementos, de 0 filas).

b) Defina el tipo del operador `addM` que permita generar matrices escalonadas no vacías de *n* columnas. Este operador junto con `emptyM` deberían permitir crear cualquier matriz escalonada de *n* columnas.

c) Construya utilizando los operadores `emptyM` y `addM` una matriz escalonada de 3 columnas de números naturales, cuyos elementos cumplan que: los elementos en la columna *i* sean todos iguales a *i*. En este parte use el tipo `nat` predefinido de Coq.