Construcción Formal de Programas en Teoría de Tipos
Primer Parcial – Octubre de 2012

Problema 1. Formalice y demuestre la validez del siguiente razonamiento en Coq, sin usar tácticas automáticas:

Hip.1 *Una base de datos no está normalizada o no presenta redundancia de información.*
Hip.2 *Una base de datos es consistente o no es útil.*

Tesis *Si una base de datos está normalizada y es útil, entonces es consistente y no presenta redundancia de información.*

Problema 2. Sea R una relación binaria sobre elementos de un conjunto U. Una relación binaria *R* entre objetos cualesquiera es *irreflexiva* si y solamente si ningún objeto está relacionado consigo mismo. *R* es *asimétrica* si y solamente si en caso de que *x* esté relacionado con *y* entonces *y* no lo está con *x*, cualesquiera sean los objetos *x* e *y*.

Pruebe en Coq, sin usar tácticas automáticas ni lógica clásica, que si *R* representa una relación *asimétrica* entonces esa relación es *irreflexiva*.

Problema 3. Pruebe el lema3 en Coq usando lógica clásica, teniendo en cuenta las siguientes definiciones:

```
Section Problema3.

        Variable C : Set.
        Variable P : C -> C -> Prop.

        Lemma L3 : (exists x y : C, P x y) \/ ~(exists x : C, P x x).

End Problema3.
```

Problema 4. Considere las siguientes declaraciones en Coq:

```
Section Problema4.
        Variable Nat: Set.
        Variable zero: Nat.
        Variable suc: Nat->Nat.
        Variable sum: Nat->Nat->Nat.
        Variable prod: Nat->Nat->Nat.
        Axiom sum0 : forall n: Nat, sum n zero = n.
        Axiom sumS : forall m n: Nat, sum m (suc n) = suc (sum m n).
        Axiom prod0 : forall n: Nat, prod n zero = zero.
        Axiom prodS : forall m n: Nat, prod m (suc n) = sum m (prod m n).
        Axiom allNat : forall n: Nat,  n = zero \/ exists m: Nat, suc m = n.
        Axiom discNat : forall n: Nat,  suc n <> zero.
```

Bajo este contexto, demuestre en Coq los siguientes lemas:

```
        Lemma L41: forall n: Nat, exists m: Nat, prod n (suc m) = sum n n.

        Lemma L42: forall m n: Nat, m <> zero -> sum n m <> zero.

        Lemma L43: forall m n: Nat, sum m n = zero -> m = zero /\ n = zero.

End Problema4.
```

Problema 5. Considere el siguiente tipo de expresiones:

```
        Parameter exp : type → Set.
```

donde `type` es un conjunto de constantes de tipo.

a) Defina el conjunto `type` cuyos elementos son las constantes *TNat* y *TBool*, que representan respectivamente el tipo de los naturales y el de los booleanos.

b) Defina el tipo de i) los operadores *NConst* y *Plus* que dados un elemento de tipo nat y dos expresiones naturales, respectivamente, retornan una expresión natural, ii) los operadores *BConst* y *And* que dados un elemento de tipo bool y dos expresiones booleanas, respectivamente, retornan una expresión booleana, y iii) un operador *Eq* que dadas dos expresiones naturales retornan una expresión booleana.

c) Defina el tipo de un operador *If*, que dada una expresión booleana y dos expresiones t1 y t2 retorne una de las dos expresiones (de acuerdo al comportamiento habitual de este operador).

<u>NOTA</u>: en este problema use los tipos `nat` y bool predefinidos de Coq cuando lo considere necesario.

Problema 6. Considere las declaraciones:

```
Variable D : Set.
Variable e : D.
Variable V : D -> D -> Prop.
Variable Q T : D -> Prop.
```

a) Complete la siguiente prueba en Coq utilizando únicamente una aplicación de la táctica `exact` con el término asociado.

```
Lemma L6a: (forall x y: D, V x y -> V y x) -> exists z : D, V e z -> V z e.
Proof.
      …
```

b) Demuestre el siguiente lema en un sólo renglón utilizando compositores (estructuradores) de tácticas:

```
Lemma L6b: (forall x:U, Q x) \/ (forall y:U, T y) → forall z:U, Q z \/ T z.
Proof.
      …
```