Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Programación Avanzada
Primer Cuatrimestre de 2017

Ejercicios de Lógica

Ejercicio1. La función *nand* puede ser definida como $p \; nand \; q = \neg(p \wedge q)$, definir esta función en Haskell sin utilizar las funciones $\neg$ y $\wedge$.

Ejercicio 2. La función $maj(x, y, z)$ retorna true ssi al menos dos de sus argumento son true, definirla en Haskell

Ejercicio 3. En Haskell un predicado sobre un tipo A es una función `p::A ->Bool`, por ejemplo:

```
even :: Int → Bool
even x = (x mod 2 == 0)
```

es un predicado sobre números cuya variable libre es x

Además podemos escribir predicados con varias variables libres, por ejemplo funciones del estilo: `p:: Int ->(a->Bool)` pueden interpretarse como predicados sobre el tipo A que tienen una variable libre de tipo entero, un ejemplo es:

```
isEven :: Int → [Int] → Bool
isEven i xs = xs.i mod 2 == 0
```

es decir, dado un indice y una lista, dice si el elemento en la posición dada es par o no.

Utilizando estas ideas podemos escribir cuantificadores en Haskell. Por ejemplo, el cuantificador universal es una función:

```
todos:: [Int]→ [a]→ (Int → [a] → Bool)→ Bool"
```

que toma una lista de indices (el rango), una lista y un predicado y dice si el predicado se cumple para todos los elementos de la lista que estén en el rango. Una posible definición es:

```
todos :: [Int] → [a] → (Int → [a] → Bool)→ Bool
todos is xs p = foldl (&&) true ys
              where ys = [p i xs | i← is ]
```

Utilizar esta función para escribir las siguientes especificaciones en Haskell:

- $\langle \forall i : 0 \leq i < \#xs : xs.i \; mod \; 2 == 0 \rangle$
- $\langle \forall i : 0 \leq i < \#xs \wedge i \; mod \; 2 == 1 : xs.i > 0 \rangle$

Ejercicio 4. Utilizando la mismas ideas del ejercicio 3, escribir los cuantificadores de sumatoria, productora y contatoria.