Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Programación Avanzada
Primer Cuatrimestre de 2021

Les recomendamos, antes de comenzar a resolver los ejercicios, repasar el teórico 5: Lógica. Además pueden leer los capítulos 2, 3 ,4 5 y 6 del Cálculo de Programas, los mismos se encuentran en la parte de Material Bibliográfico.

Ejercicios de Lógica

Ejercicio 1. 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)
```

Se puede pensar como un predicado sobre números cuya variable libre es x

Además en Haskell tenemos las siguientes funciones de alto orden:

```
and :: [Bool] -> Bool
or :: [Bool] -> Bool
```

La función `and`, dada una lista de bools, retorna `True` cuando todos sus elementos son `True`, y `False` en otro caso. La función `or` se comporta de forma dual. Con estos dos operadores y listas por comprensión podemos escribir una versión ejecutable de los cuantificadores en Haskell. Por ejemplo, el siguiente cuantificador:

$$\langle \forall i : 0 \leq i < \#xs : \text{even } xs.i \rangle$$

Puede escribirse como:

```
and $ [even $ xs !! i | i <- [0..(length xs)-1]]
```

Utilizar estas ideas para escribir los siguientes cuantificadores:

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

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