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

Trabajo Práctico

Este trabajo práctico debe ser resuelto en grupos de máximo tres personas. El proyecto podrá ser entregado cualquier fecha anterior al **Miércoles 30 de Junio.**.

El objetivo del presente proyecto es desarrollar un programa funcional simple para verificar validez y satisfacibilidad de fórmulas lógicas. Para ello consideremos el siguiente tipo definido:

```
data Formula = VAR Char | OR Formula Formula | AND Formula Formula | EQUIV Formula Formula
                        | NOT Formula deriving (Eq, Show)
```

Es decir, `Formula` define el tipo de las fórmulas proposicionales, notar que las variables son del estilo `VAR 'p'`. Además una *asignación* es un *mapeo* de valores booleanos a variables (una fila de una tabla de verdad), las mismas. Para modelar esto utilizaremos el tipos de datos `Map` definida en la libreria `Map`. Para usar dicho tipo de datos se debe importar la libreria `Data.Map` (mas detalles de esta libreria se pueden encontrar en `http://aprendehaskell.es/content/Modulos.html#data-map`).

Intuitivamente, `m :: Map k a` significa que `m` es un maping (una función finita) de un tipo con orden `k` (el tipo de las claves) a algún tipo `a`. Teniendo en cuenta esto podemos definir las asignaciones de valores de verdad a variables booleanas como:

```
type Asign = Map Char  Bool
```

Para poder realizar nuestro algoritmo necesitaremos las siguientes funciones:

- `vars:: Formula -> [Char]`, que dada un fórmula, devuelve la lista de variables en esa formula,
- `eval :: Formula-> Asign -> Bool`, que dadas una fórmula y una asignación, dice si la fórmula es verdadera o no para esa asignación.
- `asigns :: [Char] -> [Asign]`, que dada una lista de variables devuelve la lista de todas las asignaciones posibles para esas variables.
- `sat :: Formula -> Bool`, que dada una fórmula dice si es satisfacible (es decir, si hay una asignación que la hace verdadera). Una idea para implementar esta función es probar para cada posible asignación si la formula es verdadera o falsa para esa asignación.
- `valid::Formula -> Bool`, que dada una formula dice si esa formula es una tautología o no, una idea es recorrer la lista de asignaciones para las variables que ocurren en la formula, y ver que para todas ellas su evaluación de la formula devuelve verdadero.

Utilizar funciones de alto orden para la definición de `sat` y `valid`. Utilice su programa para ver que la regla dorada es una tautología, elija 5 axiomas más del libro y compruebe su validez.

Un fórmula se dice en DNF (formal normal disyuntiva) si: es una diyunción de clausulas, en donde cada clausula es una conjunción de variables proposicionales o negaciones de estas. Por ejemplo:

$$(p \wedge \neg q) \vee (\neg s)$$

se encuentra en DNF, pero:

$$\neg(p \wedge q)$$

no se encuentra en DNF.

- Implementar una función: `esDNF :: Formula -> Bool`, que dice si una fórmula está en DNF o no.