Práctica 2
Lógica computacional 2020-2
Lógica proposicional
Fecha de entrega: **domingo 1 de marzo de 2020**

En esta práctica se revisarán mecanismos para verificar si una fórmula escrita en lógica proposicional es correcta; para ello se tendrá que definir el siguiente tipo de dato para representar las fórmulas en Haskell:

```
data Prop = T | F | Var String | Neg Prop | Conj Prop Prop | Disy Prop Prop |
                    Impl Prop Prop | Equi Prop Prop deriving Eq
```

1. Interpretaciones

Como se pudo ver en clase, un estado es una asignación que toma una variable proposicional y le asigna un valor. Por ejemplo, podemos pensar en el estado $\{(p, 0), (q, 1)\}$. Una forma más compacta de representar dicho estado es utilizando una lista de variables proposicionales. Si una variable se encuentra en la lista, le asignamos 1; en otro caso, 0. De esta forma, podemos representar el estado $\{(p, 0), (q, 1)\}$ como la lista $[q]$.

De esta manera tenemos que una interpretación es una función que toma una fórmula proposicional junto con un estado y realiza la evaluación de dicha fórmula.

Por ejemplo, si tenemos el estado $\sigma = [q]$, entonces:

$$\mathcal{I}_\sigma(p \to (p \vee q)) = 0 \to (0 \vee 1) = 0 \to 1 = 1$$
$$\mathcal{I}_\sigma(p \wedge q) \vee r = (0 \wedge 1) \vee 0 = 0 \vee 0 = 0$$

Bajo esta definición, la lista del estado, son justamente sólo las variables que tienen como valor 1, y asumiendo que las variables que no aparecen en la lista tendrán un valor de 0.

La representación de estados en Haskell sería bajo el tipo

```
type Estado = [String]
```

A partir de estas definiciones realice los siguientes ejercicios:

1. Definir una función que dada una fórmula proposicional nos regrese una lista con todas las variables proposicionales que aparecen en la fórmula proposicional.

   Firma de la función:
   ```
   variables :: Prop -> [String]
   ```
   Ejemplo:
   ```
   Prelude>variables (Impl (Conj p q) p)
   ["p","q"]
   Prelude>variables Conj q (Disy r p)
   ["q","r","p"]
   ```

2. Definir la función `conjPotencia`, tal que `(conjPotencia x)` es la lista de todos los subconjuntos de x.

   Firma de la función:
   ```
   conjPotencia :: [a] -> [[a]]
   ```
   Ejemplo:
   ```
   Prelude>conjPotencia [1,2]
   [[]; [2]; [1]; [1; 2]]
   ```

3. Definir una función que dada una fórmula proposicional y un estado regrese la interpretación obtenida de la formula en dicho estado.

   Firma de la función:
   ```
   interpretacion :: Prop -> Estado -> Bool
   ```

   Ejemplo:
   ```
   Prelude>interpretacion Conj (Var "q") (Disy (Var "r") (Var "p")) ["p"]
   False
   Prelude>interpretacion Conj (Var "q") (Disy (Var "r") (Var "p")) ["p","q"]
   True
   ```

4. Definir una función que dada una fórmula proposicional, devuelve todos los estados con los que es posible evaluar la fórmula.

   Firma de la función:
   ```
   estadosPosibles :: Prop -> [Estado]
   ```

   Ejemplo:
   ```
   Prelude>estadosPosibles Disy (Var "q") (Conj (Var "r") (Var "q"))
   [[],["q"]; ["r"]; ["q","r"]]
   ```

5. Definir una función que dada una fórmula proposicional, indique si es una tautología.

   Firma de la función:
   ```
   tautologia :: Prop -> Bool
   ```

   Ejemplo:
   ```
   Prelude>tautologia Disy (Var "p") (Neg (Var "p"))
   True
   Prelude>tautologia Disy (Var "q") (Var "r")
   False
   ```

6. Definir una función que dada una fórmula proposicional, indique si es una contradicción.

   Firma de la función:
   ```
   contradiccion :: Prop -> Bool
   ```

   Ejemplo:
   ```
   Prelude>contradiccion Disy (Var "p") (Neg (Var "p"))
   False
   ```

7. Definir una función que dada una interpretación y una fórmula proposicional, verifique si esta interpretación es un modelo.

   Firma de la función:
   ```
   esModelo :: Estado -> Prop -> Bool
   ```

   Ejemplo:
   ```
   Prelude>esModelo ["r"] (Conj (Disy p q) (Disy (Neg q) r))
   False
   Prelude>esModelo ["p","r"] (Conj (Disy p q) (Disy (Neg q) r))
   True
   ```

8. Definir una función que dada una fórmula proposicional devuelva la lista de todos sus modelos; tal que $(modelos\,\varphi)$ es la lista de todas las interpretaciones de φ que son modelo.

Firma de la función:
```
modelos :: Prop -> [Estado]
```

Ejemplo:
```
Prelude>modelos (Conj (Disy p q) (Disy (Neg q) r))
[["p","q","r"],["p","r"],["p"],["q","r"]]
```

9. Definir una función que dada una fórmula proposicional φ, verifica si es válida.

   Firma de la función:
   ```
   esValida :: Prop -> Bool
   ```

   Ejemplo:
   ```
   Prelude>esValida (Impl p p)
   True
   Prelude>esValida (Impl p q)
   False
   ```

10. Definir una función que dada una fórmula proposicional φ, verifique si es insatisfacible.

    Firma de la función:
    ```
    esInsatisfacible :: Prop -> Bool
    ```

    Ejemplo:
    ```
    Prelude>esInsatisfacible (Conj p (Neg p))
    True
    Prelude>esInsatisfacible (Conj (Impl p q) (Impl q r))
    False
    ```

11. Definir una función que dada una fórmula proposicional φ, verifique sies satisfacible.

    Firma de la función:
    ```
    esSatisfacible :: Prop -> Bool
    ```

    Ejemplo:
    ```
    Prelude>esSatisfacible (Conj p (Neg p))
    False
    Prelude>esSatisfacible (Conj (Impl p q) (Impl q r))
    True
    ```

NOTA

Recuerden que deben seguir los lineamientos establecidos para la entrega de su práctica, los cuales se encuentran en la página del curso.