Universidad Nacional Autónoma de México
Facultad de Ciencias

Estructuras Discretas

Práctica 4

Javier Enríquez Mendoza **Mauricio E. Hernández Olvera**

5 de Octubre de 2018
Fecha de entrega: 19 de Octubre de 2018

Instrucciones generales

La práctica debe resolverse en el archivo `Prop.hs` y las firmas de las funciones deben ser idénticas a las que se muestran en cada ejercicio. Cada función y definición debe estar debidamente comentada con la especificación de ésta.

Se tomará en cuenta la legibilidad y el estilo del código.

Lógica Proposicional

En el archivo `Prop.hs` se encuentra la siguiente definición del tipo de dato algebraico `Prop` para definir expresiones de la lógica proposicional, así como el sinónimo `Estado` que permite representar un conjunto de estados de las variables para una interpretación.

```
-- Tipo de dato Algebraico que representa expresiones
-- de la lógica proposicional.

data Prop = Verdadero
          | Falso
          | Var String
          | Neg Prop
          | Conj Prop Prop
          | Disy Prop Prop
          | Impl Prop Prop
          | Syss Prop Prop

-- Sinónimo para representar el conjunto de Estados.

type Estado = [(String, Prop)]
```

1 Equivalencias Lógicas

Ejercicio 1.1 (1 pt.) Definir la función `eliminacion` que utiliza la regla de equivalencia de eliminación de operadores para quitar las implicaciones y equivalencias de las proposiciones.

Eliminación de operadores:

$$P \to Q \equiv \neg P \vee Q$$
$$P \longleftrightarrow Q \equiv (\neg P \vee Q) \wedge (P \vee \neg Q)$$

```
eliminacion :: Prop -> Prop
```

```
> eliminacion (Impl (Var "P") (Var "Q"))
(¬ P ∨ Q)
> eliminacion (Conj (Impl (Var "P") (Var "P")) (Var "R"))
((¬ P ∨ P) ∧ R)
```

Ejercicio 1.2 (1 pt.) Definir la función `deMorgan` utiliza las reglas de equivalencia de De Morgan para regresar proposiciones equivalentes.

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$
$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

```
deMorgan :: Prop -> Prop
```

```
> deMorgan (Neg (Conj (Var ‘‘P’’) (Var ‘‘Q’’)))
(¬ P ∨ ¬ Q)
> deMorgan (Disy (Var ‘‘P’’) (Var ‘‘Q’’))
(P ∨ Q)
```

2 Evaluación y Análisis Sintáctico de expresiones

Ejercicio 2.1 (2 pt.) Definir la función `interp` que recibe una `Prop` y un `Estado` e interpreta la proposición bajo el estado.

```
interp :: Prop -> Estado -> Bool
```

```
> interp (Impl (Var "P") (Var "Q")) [("P",Falso),("Q",Verdadero)]
True
> interp (Conj (Var "P") (Neg (Var "P"))) [("P", Falso)]
False
```

Ejercicio 2.2 (4 pt.) Definir la función `truthTable` que recibe una proposición y nos dice si es tautología, contradicción o contingencia.

Definición 2.2.1 (Tautología) Si $\mathcal{I}(P) = 1$ para toda interpretación $\mathcal{I}$, entonces P es una tautología.
Definición 2.2.2 (Contradicción) Si $\mathcal{I}(P) = 0$ para toda interpretación $\mathcal{I}$, entonces P es una contradicción.

```
truthTable :: Prop -> String
```

```
> truthTable (Impl (Conj (Impl (Var "P") (Var "Q")) (Var "P")) (Var "Q"))
Tautología
> truthTable (Conj (Var "P") (Neg (Var "P")))
Contradicción
```

Ejercicio 2.3 (2 pt.) Definir la función `correcto` que recibe una lista de `Prop` que serán las premisas y una `Prop` que será la conclusión y dice si el argumento es correcto o no.

Definición 2.3.1 (Argumento correcto) Un argumento $A_1, A_2, ..., A_n / \therefore B$ es correcto, si y sólo si suponiendo que las premisas $A_1, A_2, ..., A_n$ son verdaderas, entonces necesariamente la conclusión B también lo es.

```
correcto :: [Prop] -> Prop -> Bool
```

```
> correcto [(Var "P")] (Var "P")
True
> correcto [(Impl (Var "P") (Var "Q")), (Var "P")] (Var "Q")
True
```

Pruebas Unitarias

En el archivo `testP04.hs` se agregaron una serie de pruebas que verifican el correcto funcionamiento de cada una de las funciones de esta práctica.

Para poder correr estas pruebas, se tiene que copiar el archivo en el mismo directorio en el que se encuentre `Prop.hs`, y desde la terminal ejecutar los siguientes comandos para compilar y ejecutar las pruebas respectivamente.

```
> ghc testP04.hs
> ./testP04
```

Se mostrará en la consola los resultados de cada una de las pruebas.

Hay que notar que en esta ocasión no estamos usando la versión interactiva de GHC, como lo hemos estado haciendo.

Se recomienda no modificar el archivo `testP04.hs`.

Entrega

- La entrega se realiza mediante correo electrónico a la dirección de los ayudantes de laboratorio (javierem_94@ciencias.unam.mx y mauriciohdez08@ciencias.unam.mx).
- Es **necesario** que el correo se envíe a ambos ayudantes.
- La practica deberá ser entregada en equipos de máximo 3 personas.
- Se debe entregar un directorio `numeroCuenta_P04`, dónde `numeroCuenta` es el número de cuenta de un integrante del equipo. Dentro del directorio se debe incluir:
  - Un archivo readme.txt con los nombres y números de cuenta de los alumnos, comentarios, opiniones, críticas o ideas sobre la práctica.
  - Los archivos requeridos en la práctica. Debe enviarse código lo más limpio posible.
- Los archivos requeridos para esta práctica son: `Prop.hs`.
- El directorio se deberá comprimir en un archivo con nombre `numeroCuenta_P04.tar.gz`,dónde `numeroCuenta` es el número de cuenta de un integrante del equipo.
- Únicamente **un integrante** del equipo deberá enviar el correo con la práctica.
- El asunto del correo debe ser [ED-20191-P04].
- Se recibirá la práctica hasta las 23:59:59 horas del día fijado como fecha de entrega, cualquier práctica recibida después no será tomada en cuenta.

- **Cualquier práctica total o parcialmente plagiada, será calificada automáticamente con cero y no se aceptarán más prácticas durante el semestre.**