Práctica 01
Lógica Computacional
Universidad Nacional Autónoma de México

Mauricio Esquivel Reyes

1. Estructuras

1.1. Naturales

Consideremos la siguiente representación de los números naturales

```
data Natural = Cero | Suc Natural deriving (Eq, Show)
```

1.1.1. mayorQue :: Natural ->Natural ->Bool

Dados dos naturales nos dice si el primero es mayor que el segundo.
Ejemlos:

- Main>mayorQue Cero (Suc Cero)
  False
- Main>mayorQue (Suc Cero) Cero
  True

1.1.2. menorQue :: Natural ->Natural ->Bool

Dados dos naturales nos dice si el primero es menor que el segundo.
Ejemlos:

- Main>menorQue Cero (Suc Cero)
  True
- Main>menorQue (Suc Cero) Cero
  False

1.1.3. igual :: Natural ->Natural ->Bool

Dados dos naturales nos dice si son iguales.
Ejemplos:

- Main>igual Cero (Suc Cero)
  False
- Main>igual (Suc Cero) (Suc Cero)
  True

1.2. Lista de naturales

Consideremos la siguiente definición de las listas de naturales.

```
data ListaDeNaturales = Nil | Cons Natural ListaDeNaturales
```

1.2.1. concate :: ListaDeNaturales ->ListaDeNaturales ->ListaDeNaturales

Dadas dos listas de naturales regresar la concatenación de ambas.
Ejemplos:

- Main>concate (Cons (Suc Cero) Nil) (Cons Cero (Cons (Suc (Suc Cero)) Nil))
  Cons (Suc Cero) (Cons Cero (Cons (Suc (Suc Cero)) Nil))
- Main>concate (Cons Cero (Cons (Suc (Suc Cero)) Nil)) (Cons (Suc Cero) Nil)
  Cons Cero (Cons (Suc (Suc Cero)) (Cons (Suc Cero) Nil))

1.2.2. reversa :: ListaDeNaturales ->ListaDeNaturales

Dada una lista regresar la reversa de dicha lista.
Ejemplos:

- Main>reversa (Cons Cero (Cons (Suc (Suc Cero)) (Cons (Suc Cero) Nil)))
  Cons (Suc Cero) (Cons (Suc (Suc Cero)) (Cons Cero Nil))

2. Lógica Proposicional

Consideremos la siguiente representación de la lógica proposicional.

```
-- Tipo de dato indice
type Indice = Int

-- Tipo de dato fórmula
data PL = Top | Bot  | Var Indice
      | Oneg PL
      | Oand PL PL | Oor PL PL
      | Oimp PL PL deriving (Eq, Show)
```

2.1. hayImplicacion :: PL ->Bool

Dada una fórmula regresa un valor de verdad si hay una implicación en dicha fórmula.
Ejemplo:

- Main>hayImplicacion Oor (Var 1) (Oimp (Var 2) (Var 3))
  True

2.2. disy :: PL ->[PL]

Dada una fórmula regresar una lista con las disyunciones de dicha fórmula.
Ejemplo:

- Main>disy Oand (Oor (Var 1) Oneg $ Var 2) (Oor Bot (Var 3))

Oor (Var 1) Oneg $ Var 2, Oor Bot (Var 3)

2.3. numConj :: PL ->Int

Dada una fórmula regresar el número de conjunciones que tiene dicha fórmula.
Ejemplo:

- Main>numConj Oand (Oor (Var 1) Oneg $ Var 2) (Oand Top (Var 3))
  2

3. Sistema L de Lukasiewicz

Un sistema de deducción al estilo Hilbert para la PLI. Se utilizarán los archivos SintaxisPLI.hs, DeduccionL.hs y DeduccionLEjemplos.hs.

3.1. Definición

$$PLI ::= Bot|v < Indice > |(PLI \to PLI)$$

$$< Indice >::= [i|i \in \mathbb{N}]$$

Sea $\phi \in PLI$. La negación de ϕ se define mediante $\neg\phi = (\phi \to Bot)$

3.2. Axiomas

Axiomas para toda α, β y γ en PLI:

- L1. $\alpha \to (\beta \to \alpha)$
- L2. $(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$
- L3. $(\neg\alpha \to \neg\beta) \to (\beta \to \alpha)$

3.3. Modus Ponens

El Modus Ponens es una regla de inferencia de la lógica proposicional. Se puede resumir como: Si $\alpha \to \beta$ y α es verdad entonces se puede inferir que β también es verdad.

3.4. Deducciones en el Sistema L.

Def. Sean $\phi \in$ PLI y $\Gamma \subset$ PLI.

Decimos que ϕ se deduce de Γ en el sistema L, $\Gamma \vdash \phi$ si existe una lista finita de formulas $\gamma_1, \gamma_2, \ldots, \gamma_n \in$ PLI, tal que:

- $\gamma_n = \phi$
- Para toda $k \in 1, \ldots, n$ se cumplen:
  - $\gamma_k \in \Gamma$ (premisa)
  - γ_k es una instancia de un axioma de L.
  - Existe $i, j < k$ tales que γ_k es resultado de aplicar MP a γ_i y γ_j. (MP i,j)

3.5. Funciones

Desarrolla las siguientes funciones en donde sean necesarias.

3.5.1. esAxL1 :: PLI ->Bool

Función que nos dice si una fórmula de PLI cumple el axioma 1.
Ejemplo:

- Main>esAxL1 (Var 1) ‘Oimp‘ ((Var 2) ‘Oimp‘ (Var 1)) True

3.5.2. esAxL2 :: PLI ->Bool

Función que regresa el resultado de verificar que una fórmula de PLI cumple el axioma 2.
Ejemplo:

- Main>esAxL2 (Bot) ‘Oimp‘ ((Var 1) ‘Oimp‘ (Var 2)) False

3.5.3. esAxL3 :: PLI ->Bool

Función que decide si la fórmula dada de PLI cumple el axioma 3.
Ejemplo:

- Main>esAxL3 (((Var 1) ‘Oimp‘ Bot) ‘Oimp‘ ((Var 2) ‘Oimp‘ Bot)) ‘Oimp‘ ((Var 2) ‘Oimp‘ (Var 1))
  True

3.5.4. esAxiomaDeL :: PLI ->Bool

Función que indica si una fórmula de PLI es una instancia de los axiomas.
Ejemplo:

- Main>esAxiomaDeL (Var 2) ‘Oimp‘ ((Var 3) ‘Oimp‘ (Var 2)) True

3.5.5. esModusPonens :: PLI ->Bool

Función que recibe una tripleta de fórmulas, nos dice si la última formula es resultado de hacer MP con las anteriores.
Ejemplo:

- Main>esModusPonens (Var 1, ((Var 1) ‘Oimp‘ (Var 2)), Var 2) True

3.5.6. checkPaso

Hay que implementar los casos faltantes.

1. Prem Debe revisar que la fórmula sea parte de las premisas.
2. Ax Debe revisar que la fórmula sea una instancia de un axioma.