

Facultad de Ciencias UNAM
Lógica Computacional
Práctica 2: Lógica Proposicional

Profesor: Francisco Hernández Quiroz
Ayudante: Valeria Garcia Landa
Ayudante de laboratorio: Sara Doris Montes Incin

Entrega: 14 de febrero de 2020 antes de las 11:59 p. m.

1 Introducción

1.1 Lógica Proposicional

Consideremos la siguiente representación de la lógica proposicional

```
-- Tipo de dato indice
type Indice = Int
-- Tipo de dato fórmula
data LP = T | F | Var Indice
        | Neg LP
        | And LP LP | Or LP LP
        | Imp LP LP deriving (Eq, Show)
```

La indicación al final de los datos deriving (Show, Eq) hacen referencia a diferentes características que queremos que tenga el tipo de dato que estamos creando.

- **Show:** Es para decirle a Haskell que el tipo que estamos definiendo se puede representar con una cadena.
- **Eq:** Es porque necesitamos que los valores que puede tomar algo del tipo a definir son comparables. Es decir nos permite usar los operadores == y /= para comparar valores de este tipo.

1.2 Evaluación

Decimos que e es una evaluación si $e : X \rightarrow \{V, F\}$ y $X \subseteq \text{VarProp}$.

```
type Evaluacion = Indice -> Bool
```

Sean $e: \text{VarProp} \rightarrow \{V, F\}$ una evaluación y $\phi \in \text{PL}$, definimos la relación $e \models \phi$, usando recursión sobre la estructura de ϕ :

```
satLP :: Evaluacion -> LP -> Bool
satLP e phi = case phi of
  T -> True
  F -> False
  Var alpha -> (e alpha)
  Neg alpha -> not(satLP e alpha)
  And alpha beta -> (satLP e alpha) && (satLP e beta)
  Or alpha beta -> (satLP e alpha) || (satLP e beta)
  Imp alpha beta -> not(satLP e alpha) || (satLP e beta)
```

1.3 Modelos

m es un modelo si $m \subseteq \text{Var}$

```
type Modelo = [Indice]
```

$\text{satMod } m \phi = \text{True}$ sii $m \models \phi$

```
satMod :: Modelo -> LP -> Bool
satMod m phi = case phi of
  T -> True
  F -> False
  Var n -> elem n m
  Neg alpha -> not(satMod m alpha)
  And alpha beta -> (satMod m alpha) && (satMod m beta)
  Or alpha beta -> (satMod m alpha) || (satMod m beta)
  Imp alpha beta -> not(satMod m alpha) || (satMod m beta)
```

2 Ejercicios

1. Implementa una función recursiva que recibe una fórmula y devuelve el conjunto (lista sin repeticiones) de variables que hay en la fórmula.

Firma de la función

```
varForm :: LP -> [Indice]
```

2. Implementa una función que obtenga el conjunto potencia de un conjunto dado.

Firma de la función

```
conjuntoPot :: [t] -> [[t]]
```

3. Implementa una función que recibe una fórmula y devuelve True si la fórmula es válida y False en otro caso.

Firma de la función

`esVal :: LP -> Bool`

4. Implementa una función que recibe una fórmula y devuelve True si la fórmula es satisfacible y False en otro caso.

Firma de la función

`esSat :: LP -> Bool`

5. Implementa una función que recibe una fórmula y elimina implicaciones.

Firma de la función

`quitaImp :: LP -> LP`