Facultad de Ciencias UNAM
Lógica Computacional
Práctica 4: Sistema L de Lukasiewicz

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

Entrega: 20 de marzo de 2020 antes de las 11:59 p. m.

1 Sistema L de Lukasiewicz

1.1 Introducción

Ahora presentamos una teoría axiomática formal L para el cálculo proposicional.

1. Los símbolos de L son: $\neg$, $\to$, (,) y las letras A_i con $i \in \{1, 2, 3, 4, 5...n\}$. Los símbolos $\neg$ y $\to$ se les denomina *conectivos primitivos* y las letras A_i se les denomina *letras de declaración*
2. (a) Todas las A_i (letras de declaración) están wfs.
   (b) Si α y β están wf entonces $(\neg\alpha)$ y $(\alpha \to \beta)$ también lo están.
3. Si α, β y γ están wfs, entonces los siguientes son los axiomas de L:
   (A1) $(\alpha \to (\beta \to \alpha))$
   (A2) $((\alpha \to (\beta \to \gamma)) \to (\alpha \to \beta) \to (\alpha \to \gamma))$
   (A3) $(((\neg\alpha) \to (\neg\beta)) \to (\beta \to \alpha))$
4. La única regla de inferencia de L es $Modus\ Ponens$: β es una consecuencia de de α y $\alpha \to \beta$. Abreviaremos la aplicación de está regla usando MP.

1.2 Definición

Como se mencionó en la sección anterior, en el sistema L solo tenemos como operadores a $\neg$ y $\to$. Por lo que definimos un nuevo tipo de dato:

$$PLI ::= F \mid v <Indice> \mid (PLI \to PLI)$$
$$<Indice> ::= [i \mid i \in N]$$

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

2 Deducción 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 $f_1, f_2, .., f_n \in$ PLI, tal que:

- $f_n = \phi$
- Para toda k $\in$ 1,...,n se cumplen:
  - $f_k \in \Gamma$ (premisa)
  - f_k es una instancia de un axioma de L.
  - Existe i, j $<$ k tales que f_k es resultado de aplicar MP a f_i y f_j . (MP i,j)

3 Ejercicios

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

   ```
   esAx1:: LP -> LP
   ```

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

2. Función que nos dice si una fórmula de PLI cumple el axioma 2

   ```
   esAxL2 :: PLI ->Bool
   ```

   - Main > esAxL2 (F) ‘Imp‘ ((Var 1) ‘Imp‘ (Var 2))
     False

3. Función que nos dice si una fórmula de PLI cumple el exioma 3

   ```
   esAxL3 :: PLI ->Bool
   ```

   - Main > esAxL3 (((Var 1) ‘Imp‘ F) ‘Imp‘ ((Var 2) ‘Imp‘ F)) ‘Imp‘ ((Var 2) ‘Imp‘ (Var 1))
     True

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

   ```
   esAxiomaDeL :: PLI ->Bool
   ```

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

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

5. ```
   esModusPonens :: (PLI,PLI,PLI) ->Bool
   ```

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

6. checkPaso

   Hay que implementar los casos faltantes.

   (a) Prem Debe revisar que la fórmula sea parte de las premisas.

   (b) Ax Debe revisar que la fórmula sea una instancia de un axioma
   ```