Lógica computacional: Actividad de laboratorio III

Favio E Miranda Perea Javier Enríquez Mendoza José Manuel Madrigal Ramírez

Fecha de entrega — 23 de Septiembre, 2022

Instrucciones

1) Anteriormente definimos el tipo `Clausula` que consistía en una lista de literales:

```
type Clausula = [Literal]
```

A partir de ella vamos a definir una función que transforme nuestras fórmulas del tipo `Prop` en su forma normal conjuntiva a listas de Cláusulas. Entendidas estas últimas como listas de literales:

```
clausulasFNC ::  Prop -> [Clausula]
```

Aquí algunos ejemplos de su uso:

```
*Main>clausulasFNC (VarP 1)
[[v1]]
*Main>clasusulasFNC (And (Or (Neg (VarP 1)) (VarP 2)) (VarP 3))
[[(~v1), v2], [v3]]
```

Pista: Podemos hacer una función auxiliar que transforme las cláusulas del tipo `Prop` a listas de literales, y después mandar a llamar esta función desde `clausulasFNC`.

2) Ahora definiremos una función que verifique si una interpretación dada es modelo de un conjunto de cláusulas:

```
esModeloClausulas ::  Estado -> [Clausula] -> Bool
```

Anteriormente definimos el tipo `Estado` como una lista de `Prop`, en la que aparecían variables que asumíamos como verdaderas. Veamos algunos ejemplos de la aplicación de esta función:

```
*Main>esModeloClausulas [(VarP 1),(VarP 2)] [[(VarP 1),(VarP 2)], [(VarP 1)]]
True
*Main>esModeloClausulas [(VarP 1),(VarP 2)] [[(VarP 1), (Neg (VarP 2))], [(VarP 3)]]
False
```

3) Creen un archivo readme.txt en el que incluyan sus nombres y un breve párrafo mencionando las dificultades que surgieron durante la implementación así como comentarios que consideren pertinentes.

4) Envíen un archivo comprimido que contenga su script y el archivo readme.txt a mi correo: *jose.manuel.madrigal.ramirez@gmail.com* con el siguiente formato:

Practica1-Apellido1Nombre1Apellido2Nombre2.zip

Por ejemplo, si su equipo esta conformado por Alan Turing y Ada Lovelace su archivo debería llamarse:

Practica1-TuringAlanLovelaceAda.zip