UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Integrantes:
Marco Silva Huerta
Adrián Aguilera Moreno

Lógica Computacional

Práctica 2

1. `interp` : ésta función emplea en su caso base [de la recursión] a la función básica `elem` que implementa el `PRELUDE` de Haskell.

   Todos los demás son casos recursivos de los 5 conectivos en PROP.

2. `estados` : ésta función emplea a las funciones 3 y 4 en su implementación.

3. `vars` : se obtienen las posibles variables en la proposición y se concatenan.

4. `subconj` : Por medio de una lista definida por comprensión [donde la cola es el conjunto potenecia de la cola de la lista pasada como parámetro] se define una lista y se le concatena el conjunto potencia de su cola.

5. `modelos` : por medio de una lista por comprensión se definen las interpretaciones de los estados [los que sean correctos].

6. `tautología` : Verifica que los estados y los modelos de una proposición sean los mismos.

7. `satisfacen` : verifica que la interpretación sea `True`.

8. `satisf` : si los modelos no son una lista vacia [esto es, que haya al menos un modelo], entonces la proposición es satisfacible. En otro caso, no se encontraron modelos y no es satisfacible.

9. `insatisfacen` : ésta es la negación de la función 7.

10. `contrad` : ésta es la negación de la función 8.

11. `equiv` : basta ver que los estados de ambas proposiciones sean los mismos.

Para las funciones: `elimEquiv, elimImpl, deMorgan`, se tienen los casos recursivos y los casos base, pero se tiene el cuidado de realizar los cambios a mano en cuando corresponda: por ejemplo, en las leyes de De Morgan se revisa la negación de la conjunción y de la disyunción, y es aquí donde se realiza el cambio de conjunción a disyunción [disyunción a conjunción] y se niegan ambas proposiciones.
Las demás ya no las pudimos hacer, no nos interpretó la de `consecuencia` :(, dísculpa por entregar hasta ahora, no nos vuelve a pasar.