Estructuras Discretas 2021-2
Práctica 6, 1° parte: Fórmulas Proposicionales

Facultad de Ciencias, UNAM

Fecha de entrega: 17 de diciembre del 2021

Resuelva los siguientes ejercicios, la práctica es en **parejas**. En el archivo `formulas.py` se encuentra la firma correspondiente para cada función y donde tiene que completar con código. Después de haber resuelto todo, puede ejecutar dentro de la carpeta el comando `python3 -m unittest -v test_formulas.py` para asegurarse que la salida de cada función es correcta, es importante mencionar que esto no asegura que esté bien hecho, por ejemplo, que no haya usado funciones ya definidas en `Python` que resuelvan directamente el problema pero podrá darse cuenta si alguna función es incorrecta.

Para esta práctica **todas las funciones deben ser recursivas**, de otro modo no se considerarán correctas.

Para la entrega, siga las especificaciones publicadas en `Classroom`.

1. Defina la función recursiva `repr` que devuelve la representación en cadena, legible para humanos, de una Fórmula proposicional.

   - **Variables:** su representación será xn, donde n corresponde al identificador de la variable según el constructor.

     ```
     #creo variable
     >>x1 = Formula(1)
     >>repr(x1)
     x1
     ```

   - **Negación:** sea f_1 la fórmula que es negada, su representación será $(\neg f1)$

     ```
     #creo variable
     >>x1 = Formula(1)
     #niego x1
     >>f1 = Formula(x1,'N')
     #obtengo representación legible
     >>repr(f1)
     (¬x1)
     ```

   - **Conjunción:** sean f_1 parte izquierda y f_2 parte derecha de la fórmula de una conjunción f_3, su representación esperada es $(f_1 \wedge f_2)$, así como se muestra a continuación:

     ```
     #creo dos fórmulas:
     >>var1 = Formula(1)
     >>f2 = Formula(2)
     >>f1 = Formula(var1,'N')
     #obtengo la conjunción de f1 y var2
     >>f3 = Formula(f1, 'C', f2)
     #obtengo su representación
     >>repr(f3)
     ((¬x3) ∧ x2)
     ```

   - **Disyunción:**sean f_1 parte izquierda y f_2 parte derecha de la fórmula de una disyunción, su representación esperada es $(f_1 \vee f_2)$. Por ejemplo, tomando el código anterior, si f_3 es ahora una disyunción:

     ```
     #f3 como disyunción
     >>f3 = Formula(f1, 'D', f2)
     #obtengo su representación
     >>repr(f2)
     ((¬x3) ∨ x2)
     ```

   - **Implicación:** sean f_1 parte izquierda y f_2 parte derecha de la fórmula de una implicación, su representación esperada es $(f_1 \rightarrow f_2)$. Ahora,si f_3 es una implicación, tenemos que:

```
#f3 como implicación
>>f3 = Formula(f1, 'I', f2)
#obtengo su representación
>>repr(f3)
((¬x3) → x2)
```

- **Equivalencia:** sean f_1 parte izquierda y f_2 parte derecha de la fórmula de una equivalencia, su representación esperada es $(f_1 \leftrightarrow f_2)$. Ahora,si f_3 es una implicación, tenemos que:

```
#f3 como implicación
>>f3 = Formula(f1, 'I', f2)
#obtengo su representación
>>repr(f3)
((¬x3) ↔ x2)
```

2. Defina la función recursiva `lista_variables` que devuelve una lista con las variables que aparecen en una fórmula, en orden. Tome en cuenta que las variables son enteros, por ejemplo:

```
#definimos la fórmula (x2 ∨ (x4 ∧ x1))
>>var1 = Formula(4)
>>var2 = Formula(1)
>>var3 = Formula(2)
>>for1 = Formula(var1,'C', var2)
>>for2 = Formula(var3,'D',for1)
#obtenemos la lista de variables de la fórmula 1
>>for1.lista_variables()
[1,4]
#obtenemos la lista de variables de la fórmula 2
>>for2.lista_variables()
[1,2,4]
```

3. Defina la función recursiva `ultima_variable` que devuelve la última variable que aparece en la Formula. Se considerará el orden de las variables de menor a mayor, por ejemplo, si tenemos la fórmula $((x1 \wedge x2) -> x4)$, sabemos que la lista de variables es $[1, 2, 4]$ y entonces la última variable es aquella que tiene como identificador 4

```
#otro ejemplo de ultima_variable
>>for1 = Formula(var1,'C', var2) # (x4 ∧ x1)
>>for2 = Formula(var3,'D',for1)  #(x2 ∨ (x4 ∧ x1))
#obtenemos última variable de la fórmula 1:
>>for1.ultima_variable()
4
#obtenemos última variable de la fórmula 2:
>>for2.ultima_variable()
4
```

4. Defina la función recursiva `numero_conectivos` que devuelve el número de conectivos que aparecen en la Formula. Por, ejemplo, la salida esperada de `for2` (mismo que el ejemplo del ejercicio anterior), es 2.

5. Defina la función recursiva `evalua` que devuelve el valor de verdad de la fórmula bajo una asignación dada, que recibe como entrada en la forma de una lista de booleanos. Para esto, deberán tomar en cuenta las tablas de verdad de la conjunción, disyunción, implicación, equivalencia y negación. Por ejemplo:

```
>> x = Formula(1) #x1
>> y = Formula(2) #x2
>> conjuncion = Formula(x, 'C', y)
>> asignacion = [1,1]
#resultado cuando x1 y x2 se evaluán a verdadero:
1 #verdadero
```

O, si tomamos nuevamente el ejemplo para $(x2 \vee (x4 \wedge x1))$ y la asignación $[1, 0, 0]$:

```
>>for2 = Formula(var3,'D',for1)  #(x2 ∨ (x4 ∧ x1))
>>for2.evalua([1,0,0])
0 #falso
```

6. Defina la función `aplana` que devuelva una lista con la versión aplanada del árbol de sintáxis de la fórmula.

```
# Creación de fórmulas
>>> f1 = Formula(1)
>>> f2 = Formula(2)
>>> f4 = Formula(4)
# Conjunción de f1, f2 = f3
>>> f3 = Formula(f1, 'C', f2)
# Bicondicional de f3, f4 = f5
>>> f5 = Formula(f3, 'B', f4)
# Aplanado
>>> f5.aplana()
[x1, (x1 ∧ x2), x2, ((x1 ∧ x2) ↔ x4), x4]
```

7. Defina la función `aplana_sin_variables` que devuelva una lista con la versión aplananada del árbol de sintaxis de la fórmula, sin las hojas.

```
# Creación de fórmulas
>>> f1 = Formula(1)
>>> f2 = Formula(2)
>>> f4 = Formula(4)
# Conjunción de f1, f2 = f3
>>> f3 = Formula(f1, 'C', f2)
# Disyunción de f3, f4 = f5
>>> f5 = Formula(f3, 'D', f4)
# Aplanado
>>> f5.aplana_sin_variables()
[(x1 ∧ x2), ((x1 ∧ x2) ∨ x4)]
```