LPInterpreter

Yura Hernández

***Explicación parte A* Collazos**

En esta parte contamos con tres funciones y dos diccionarios globales.

Los diccionarios son:
- values
- op

Diccionario values:
Este diccionario guardará los valores de verdad para cada átomo asignado al inicio del programa.

Diccionario op (Operators):
Este diccionario guarda los símbolos de & y | junto con su función.

Las funciones son:
- String_to_list
- val
- main

Funcion string_to_list:

Entrada: String
Salida: Lista
Esta función recibe un string y devuelve una lista sin contar los espacios.

Función val:

Entrada: Lista a evaluar, índice de comienzo, índice del final
Salida: Bool
Esta función evalúa una lista con elementos de la lógica proposicional y devuelve su valor de verdad.

Explicación detallada: La función val es una función recursiva (Se llama a sí misma) recibe de entrada una lista con los elementos de una expresión de lógica proposicional, un índice de comienzo y un índice final.

La función comienza evaluando si el índice de inicio y final es igual, si es así, devolverá el valor de verdad asignado a este átomo, si no es así verificara si el índice de inicio es un operador **not (!)**, si lo es, la función se llamara a sí misma negando el valor que regrese, de lo contrario si el valor de inicio es un **paréntesis abierto** y el valor de final es un **paréntesis cerrado** se iniciara un ciclo desde el valor de índice inicial más (+) uno (1) hasta el final, buscando un operador lógico **(Valor en diccionario op)** en medio de la expresión (sin paréntesis de por medio).

Una vez haya encontrado el operador lógico en el medio de la operación se almacenará el índice en la variable lop **(lógical operator)** y se evaluaran los resultados de las expresiones por medio de diccionarios teniendo como entradas a la función val, la función val tendrá de entradas las siguientes:

Expresión de la izquierda:
Lista completa con los elementos de lógica proposicional **(LP)**, valor de índice inicial más uno **(start + 1)**, valor de índice del operador del medio menos 1 **(lop - 1)**

Expresión de la derecha: Lista completa con los elementos de lógica proposicional **(LP)**, valor de índice del operador del medio más uno **(lop + 1)**, valor del índice final menos 1 **(end - 1)**

Esto evaluara las expresiones internas de la expresión principal, volviendo a la evaluación inicial.

Funcion main:

La función main se encarga de leer los datos de entrada y establecer los valores de verdad para cada átomo en la expresión de lógica proposicional.

Explicación parte A Yura

En esta parte contamos con 4 funciones, dos diccionarios globales y una clase.

Los diccionarios son:
values
op

Diccionario values:

Este diccionario guardará los valores de verdad para cada átomo asignado al inicio del programa.

Diccionario op (Operators):

Este diccionario guarda los símbolos de & y | junto con su función.

Clase node

La clase node cuenta con 3 atributos:

- right
- center
- left

y un método llamado **true_value**

- Lo que hace es evaluar lo que se tiene al lado izquierdo y derecho con el operador que este situado en el centro.

Funcion no_space_str:

Entrada: str
Salida: str
Esta función recibe un string con espacios y lo devuelve sin ellos.

Funcion valid_LP:

Entrada: LP
Salida: bool
Esta función recibe la expresión de lógica proposicional y la evalúa usando el método de pila, adicioné como requisito que en los paréntesis haya un operador en la mitad.

Funcion val:

Entrada: LP, start, end
Salida: bool

La función separa la expresión de lógica proposicional dependiendo de su punto de partida, comenzando con el operador negación y luego con el operador en el medio, para luego devolver el valor de verdad de toda la expresión.

Explicación detallada: La función val recibe como entrada una expresión de lógica proposicional, su posición inicial y su posición final. Primero se evalúa que hay en su posición inicial, si es un signo de not **(!)** negara todo lo que hay al frente de él entrando otra vez a la misma función, pero con el punto de partida con un más uno **(+1)** y activando la flag negation para saltar el siguiente proceso. Si el punto de partida no es un operador not comenzará el proceso de encontrar el índice del operador central, y una vez lo haya encontrado se creará un nodo el cual tendrá en su lado izquierdo el valor de la expresión lógica desde el punto inicial hasta el valor de la posición del operador central menos uno **(-1)**, en la posición central se pondrá el operador y al lado derecho tendrá el valor de la expresión lógica desde la posición del operador central más uno **(+1)** hasta el final. Una vez creado el nodo dependiendo de si la expresión que se encuentra en los extremos es un átomo se convertirá en su valor de verdad, si no la expresión entrará otra vez a la función val. Una vez haya vuelto, la variable ans tomará el valor del método **true_value** y lo retornará.

Funcion main:

La función main se encarga de leer los datos y recibir el valor de la función **valid_LP** para continuar con la evaluación de las expresiones de lógica proposicional.

Explicación parte B

En esta parte contamos con 2 funciones e importamos las funciones de la parte A (val y string_to_list) junto con el diccionario values

Funcion prove_val:

Entrada: Lista con elementos de lógica proposicional (LP)
Salida: int
Esta función se encarga de evaluar si la expresión dada es una tautología, una contradicción o una contingencia por método de fuerza bruta.

Explicación detallada:
Esta función tendrá ciclos anidados con una lista de o y 1, alternando así para cada átomo (átomos ya preestablecidos) y probará cada uno de sus valores de verdad mediante la función ya creada en la parte A (función val), almacenando estos valores en una lista. Cuando ya haya probado todos los valores, se hará un conteo de los valores retornados. En caso de que se encuentren solo unos (1) se devolverá 1 mostrando que es una tautología, si solo encuentra ceros (0) devolverá 0 mostrando que es una contradicción, pero si encuentra unos y ceros (1 y 0) se devolverá -1 mostrando que es una contingencia.

Funcion main:

La función main se encarga de leer los datos de entrada y llamar la función prove_val.