UNIVERSIDAD DEL VALLE DE GUATEMALA
Teoría de la Computación

Proyecto 2- Gramáticas y Funciones Lambda

Andrea María Paniagua Acevedo - 18733
Javier Emilio Alvarez Cifuentes - 18051
Javier André Salazar De León - 18764

- Diseño de la aplicación:

  Para las funciones del algoritmo CYK se tuvieron que crear 2 listas para almacenar los datos terminales y no terminales del lenguaje. Además de un diccionario para almacenar la gramática. Para el parseo, se busca el input de un string, se analiza el largo del string y se crea una tabla con los elementos mediante un elemento de tipo set. Se rellena la tabla y se pasa por cada uno de los datos en el input para ver si la palabra puede ser formada por las reglas dadas por la gramática. En caso de ser posible, se devuelve un texto que lo afirma, en el caso contrario se despliega un texto negándolo.

  Para las funciones lambda se utiliza la palabra reservada lambda, la cual nos permite crear operaciones dentro de python, y utilizando esto y una función predefinida se realizan las diferentes funciones de los números, las cuales fueron hechas tomando en cuenta cuál sería el rol del número que representan dentro de una función. Estas funciones fueron diseñadas sin el uso de números convencionales, y en base a esto se modelan las diferentes operaciones que se realizan dentro del programa.

- Discusión:

  La complejidad de la implementación del algoritmo provino de como ir analizando las reglas dadas por la gramática e ir comparando con los elementos disponibles en la tabla. Ya que se requerían de varias verificaciones iterativas en las listas y el diccionario del que disponemos. Lo más complicado fue encontrar la forma de recorrer la tabla en diagonal, además de ir verificando a la vez que se recorre la tabla.

  La complejidad de la aplicación de los números lambda proviene principalmente de lo abstracto del concepto, ya que es algo bastante difícil de visualizar, cosa que dificulta la comprensión para la realización de las funciones lambda, pero se logra terminar y realizar dichas funciones sin utilizar números, solo se utilizan funciones para realizar las funciones principales.

También se programa las funciones alpha y beta, las cuales tiene un funcionamiento distinto de las funciones lambda, ya que estas funciones utilizan números comunes, a diferencia de los números lambda que solo son una representación de lo que ocurre cuando se utiliza el número dentro de una función común.

- Ejemplos:

Algoritmo CYK:

```
/bin/python /home/zombiewafle/Descargas/Proyecto2/cyk_main.py
[zombiewafle@G3 Proyecto2]$ /bin/python /home/zombiewafle/Descargas/Proyecto2/cyk_main.py
Time taken: 0 0.058ms
        the             cat             drink           the             beer
1       {'Det'}         {'N'}   _               {'Det'}         {'N'}
2       {'NP'}  _               _               {'NP'}
3       _               _               _
4       _               _
5       _
The input does not belong to this context free grammar!
[zombiewafle@G3 Proyecto2]$
```

```
/bin/python /home/zombiewafle/Descargas/Proyecto2/cyk_main.py
[zombiewafle@G3 Proyecto2]$ /bin/python /home/zombiewafle/Descargas/Proyecto2/cyk_main.py
Time taken: 0 0.023ms
       hello           its             me
1      _               _               _
2      _               _
3      _
The input does not belong to this context free grammar!
[zombiewafle@G3 Proyecto2]$
```

```
/bin/python /home/zombiewafle/Descargas/Proyecto2/cyk_main.py
[zombiewafle@G3 Proyecto2]$ /bin/python /home/zombiewafle/Descargas/Proyecto2/cyk_main.py
Time taken: 0 0.077ms
        the             dog             eats            the             soup
1       {'Det'}         {'N'}   {'VP', 'V'}     {'Det'}         {'N'}
2       {'NP'}  _               _               {'NP'}
3       {'S'}   _               {'VP'}
4       _               _
5       {'S'}
The input belongs to this context free grammar!
[zombiewafle@G3 Proyecto2]$
```

Funciones lambda:

Números lambda:

```
Mostrando cada numero lambda
zero(f)(0)  =  0
one(f)(0)  =  1
two(f)(0)  =  2
three(f)(0)  =  3
four(f)(0)  =  4
five(f)(0)  =  5
six(f)(0)  =  6
seven(f)(0)  =  7
eight(f)(0)  =  8
nine(f)(0)  =  9
```

Sucesor de los números lambda:

```
Mostrando cada numero lambda
zero(f)(0)  =  0
one(f)(0)  =  1
two(f)(0)  =  2
three(f)(0)  =  3
four(f)(0)  =  4
five(f)(0)  =  5
six(f)(0)  =  6
seven(f)(0)  =  7
eight(f)(0)  =  8
nine(f)(0)  =  9
```

Suma:

```
Mostrando la funcion para la suma
suma(six)(nine)(f)(0)  =  15
```

Multiplicación:

```
Mostrando la funcion para el producto
producto(six)(nine)(f)(0)  =  54
```

Potencia (El primer número es el exponente y el segundo es la base):

```
Mostrando la funcion para la potencia
potencia(three)(two)(f)(0)  =  8
```

Función alfa:

```
Ingrese el numero para la funcion alpha: 15
alpha(x)  =  16
```

Función beta:

```
Ingrese el numero para la funcion beta: 15
beta(x)  =  30
```

- Referencias:

Bach, J. (2018), The Lambda Calculus for Absolute Dummies (like myself). Extraído de: http://bach.ai/lambda-calculus-for-absolute-dummies/