INSTITUTO POLITECNICO NACIONAL

Escuela Superior de Cómputo

PRACTICA 8

Autor:
Gomez Hernandez Alan Javier

27 diciembre, 2021|

GRAMÁTICA NO AMBIGUA

RESUMEN

Elaborar un programa para procesar la gramática no-ambigua: $B->(RB|\epsilon R->)|(RR$ **En donde B es el símbolo inicial y R genera cadenas que tienen un paréntesis derecho más que uno izquierdo.** $Sinecesitamosexpandir Bentoncesusamos B->(RBsielsiguientesmboloes)yelfinal$. **Si necesitamos expandir R, usamos R ->) si el siguiente símbolo es ")" y (RR si es "(" 2 El debe contener un modo manual y automático, en el modo manual el usuario ingresara una cadena de paréntesis y sera procesada hasta llegar a una cadena final que nos indicara si la cadena esta balanceada o no**

- **La cadena puede ser ingresada por el usuario o generada automáticamente.**
- **Mandar a un archivo y/o en pantalla la evaluación de la gramática imprimiendo el símbolo que está evaluando, indicando que producción se aplicó y la cadena generada.**
- **La longitud máxima será manejar cadenas de 1,000 caracteres.**

INTRODUCCION

En este problema se debe de construir una única derivación por la izquierda para una cadena de paréntesis balanceada. Usando las siguiente GIC. $B->(RB|\epsilon R->)|(RR$ Las derivaciones no son necesariamente únicas, incluso aunque la gramática no sea ambigua; no obstante, en una gramática no ambigua, tanto las derivaciones más a la izquierda como las derivaciones más a la derecha serán únicas. Vamos a considerar únicamente las derivaciones más a la izquierda y extenderemos el resultado a la derivaciones más a la derecha. Por lo que una gramática ambigua es una gramática libre de contexto para la cual existe una cadena que puede tener más de una derivación a la izquierda o árboles de análisis. La gramática no ambigua es una gramática libre de contexto para la cual cada cadena válida tiene una derivación única a la izquierda o un árbol de análisis.

DESARROLLO

0.1 PARSEO (CODIGO)

```
# -*- coding: utf-8 -*-
from __future__ import print_function

def proceso(cadena):
    derivacion = 'B'
    archivo = open('historia-parentesis.txt', 'w')
    cadena += ' '
    print('Cadena: ', cadena)
    archivo.write('Cadena: %s'%cadena)
    continuar = True
    for simbolo in cadena:
        if not continuar:
            break
        print(derivacion, end = '\t')
        archivo.write('\n%s   ' %derivacion)
        for paso in derivacion:
            if paso == 'B':
                if simbolo == '(':
                    derivacion = derivacion.replace('B', '(RB', 1)
                    print('Regla usada: B->(RB')
                    archivo.write('Regla usada: B->(RB')
                    break
                elif simbolo == ' ':
                    derivacion = derivacion.replace('B', '', 1)
                    print('Regla usada: B->e')
                    archivo.write('Regla usada: B->e')
                    break
                elif simbolo == ')':
                    continuar = False
                    break
            elif paso == 'R':
                if simbolo == ')':
                    derivacion = derivacion.replace('R', ')', 1)
                    print('Regla usada: R->)')
                    archivo.write('Regla usada: R->)')
                    break
                elif simbolo == '(':
                    derivacion = derivacion.replace('R', '(RR', 1)
                    print('Regla usada: R->(RR')
                    archivo.write('Regla usada: R->(RR')
                    break
                elif simbolo == ' ':
```

```
                    continuar = False
                    break
    archivo.write('\nFinal: %s' %derivacion)
    print('\nFinal: ', derivacion)
    if paso == 'B' and simbolo == ' ':
        print('\nCadena balanceada')
        archivo.write('\nCadena balanceada')
    else:
        print('\nCandena no balanceada')
        archivo.write('\nCadena no balanceada')
    archivo.close()
```

0.2 Prac-gram-NA (CODIGO)

```
# -*- coding: utf-8 -*-
from __future__ import print_function
from parseo import proceso
import random

separador = '°'*50
def iniciar():
    continuar = True
    while continuar:
        opcion = imprimir_menu()
        if opcion == 1:
            entrada_consola()
        elif opcion == 2:
            ejecutar_random()
        else:
            break
        print('=' * 100)
        opcion = input("¿Una ves mas? [s/n]: ")
        if opcion.lower() != 's':
            continuar = False

    print('Saliendo del programa...')

def imprimir_menu():
    print("\nBIENVENIDO AL PROGRAMA DE GRAMATICA NO AMBIGUA \n")
    print('\n\n%sMenu%s' % (separador, separador))
    print("""
        1.- Ingresar la cadena (Manual)
        2.- Aleatorio (Automatico)
        3.- Salir
    """)
    try:
```

```
        opcion = int(input("Selecciona una opcion valida: "))
        return opcion
    except Exception as e:
        print('Error ', e)
        return 0

def entrada_consola():
    texto = input("Escribe la cadena de parentesis: ")
    proceso(texto)

def ejecutar_random():
    i = 0
    longitud_random = random.randint(1, 1000)
    cadena = ''
    while i < longitud_random:
        cadena += random.choice(['(', ')'])
        i += 1

    print("La cadena es: ", cadena)
    proceso(cadena)

iniciar()
```

0.3 FUNCIONAMIENTO Y PRUEBAS

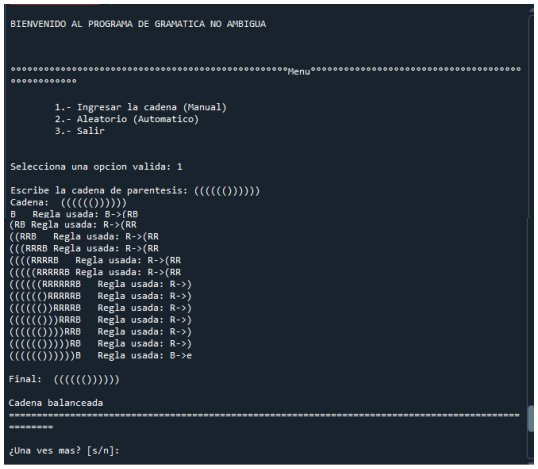

Figure 1: Menu y primera opcion de manera manual ingresamos una cadena valida y automaticamente el programa realiza la gramatica no ambigua con las reglas y lo manda al bloc de notas, con cadena balanceada

Figure 2: Archivo donde se guardo como se hizo el proceso para ver que es valida la cadena y la regla usada, como tambien como queda al final

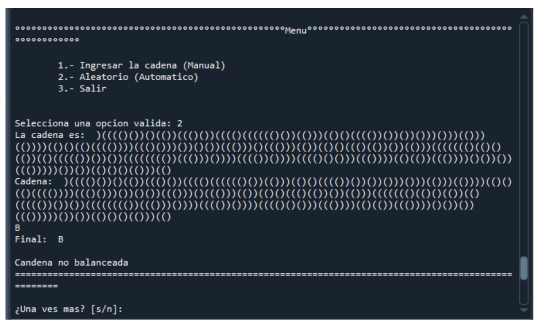

Figure 3: Menu automatico, se realizan las reglas para determinar la gramatica no ambigua y si es valida o no, segun el caso mostrado, si es balanceada o no

1 CONCLUSION

La gramática libre de contexto puede ser ambigua o no ambigua. La diferencia entre la gramática ambigua y no ambigua es que la gramática ambigua es una gramática libre de contexto para la cual existe una cadena que puede tener más de una derivación más a la izquierda mientras que una gramática no ambigua es una gramática libre de contexto para la cual cada cadena válida tiene una derivación única a la izquierda. Ahora bien para finalizar el unico problema que se pudo obtener fue la utilizacion de simbolos para obtener la regla correspondiente definida dentro de los parametros en las gramaticas no ambiguas.

References

[1] J. E. Hopcroft, R. Motwani, and J. D. Ullman, Introduccion a La Teoria De Automatas, Lenguajes Y Computacion. Addison-Wesley, 2007.

[2] J. D. Ullman, "Finite Automata." http://infolab.stanford.edu/ ullman/ialc/spr10/slides/fa1.pdf, 2010. [Consultado: 2021-12-20].

[3] M. (1984). Propedéutico: Teoría de Automatas y Lenguajes Formales Gramáticas libres de contexto y lenguajes. Isis, 75(2), 396–397. https://doi.org/10.1086/353509