Instituto Politécnico Nacional
Escuela Superior de Cómputo
Teoría Computacional
Práctica 8 : Gramática.
Alumno: AbrMa
Profesor: Jorge Luis Rosas Trigueros
Realización: 03/05/2019
Entrega: 10/05/2019

1. Marco teórico.

Una *gramática regular* G es una 4-tupla $G = (\Sigma, N, S, P)$, donde Σ es un alfabeto, N es una colección de símbolos no terminales, S es un no terminal llamado *símbolo inicial*, y P es una colección de reglas de sustitución, llamadas *producciones*, y que son de la forma $A \to w$, donde $A \in N$ y w es una cadena sobre $\Sigma \cup N$ que satisface los siguiente:

1. w contiene un no terminal como máximo.
2. Si w contiene un no terminal, entonces es el símbolo que está en el extremo derecho de w.

El *lenguaje generado* por la gramática regular G se denota por $L(G)$. Por ejemplo, considérese la gramática regular $G = (\Sigma, N, S, P)$, donde

$$\Sigma = \{a, b\}$$
$$N = \{S, A\}$$
$$P : S \to bA$$
$$A \to aaA|b|\varepsilon$$

Observése que $L(G)$ contendrá todas las cadenas de la forma $ba^{2n}byba^{2n}$. Es decir, $L(G))b(a^2)^*(b \cup \varepsilon)$. Se puede demostrar, por inducción sobre n, que todas las cadenas de la forma $ba^{2n}byba^{2n}$ están en $L(G)$ y, por inducción sobre la longitud de una derivación, se demuestra que $L(G)$ está contenido en $b(a^2)^*(b \cup \varepsilon)$.

De la definición se deduce que el lado derecho de una producción es una cadena de $\Sigma^*(N \cup \varepsilon)$. Obsérvese que ε puede ser el lado derecho de una producción. En el ejemplo precedente, la producción $A \to \varepsilon$ ahora con la generación de una cadena ya que se 'borra' el no terminal A. Dado que las producciones emparejan no terminales de N con cadenas de $\Sigma^*(N \cup \varepsilon)$, conviene representarlas como pares ordenados de $N \times \Sigma^*(N \cup \varepsilon)$. Por lo tanto, el par (x, y) de $N \times \Sigma^*(N \cup \varepsilon)$ representa la producción $x \to y$. Las producciones de P del ejemplo anterior se podrían representar mediante

$$P = \{(S, bA), (A, aaA), (A, b), (A, \varepsilon)\}.$$

2. Desarrollo de la práctica.

- Obtener una gramática regular para $(a^*b \cup b^*a)^*$ (por la derecha).

  1. Importamos las librerías de `PLY`
     ```
     import ply.lex as lex
     import ply.yacc as yacc
     ```
  2. Definimos nuestros tokens
     ```
     tokens = ('a', 'b')
     t_a = r'a'
     t_b = r'b'
     ```
  3. Definimos una función para detectar errores en la lectura de la cadena, por si se encuentra un símbolo que no pertenece a G
     ```
     def t_error(t):
     ..print(''Símbolo ilegal '%s' ''% t.value[0]) )
     ..t.lexer.skip(1)
     ```
  4. Creamos
     ```
     lex.lex()
     ```
  5. Definimos la gramática

$$P : S \to aS|bS|\varepsilon$$

de la siguiente manera :

```
def p_S(p):
..''' S : S a
.......| S b
.......| empty '''
..pass
```

6. Definimos una función para reconocer a ε

```
def p_empty(p):
..''' empty : '''
..pass
```

7. Definimos una función para indicarnos si una cadena ingresada no pertenece al lenguaje.

```
def p_error(p):
..global s
..if p:
....print('' Error de sintaxis en '%s'" % p.value)
print(s,"no está en el lenguaje")
..else:
....print(''Error de sintaxis en EOF'')
....print(s,"no está en el lenguaje")
```

8. Creamos

```
yacc.yacc()
```

9. Creamos un ciclo para leer la cadena ingresada y evaluarla

```
while 1:
.. try:
....s = input('>')
..except EOFError:
....break
..if not s: continue
.. t = yacc.parse(s)
```

10. Ejecutamos el script e ingresamos cadenas fig. 1 (a).

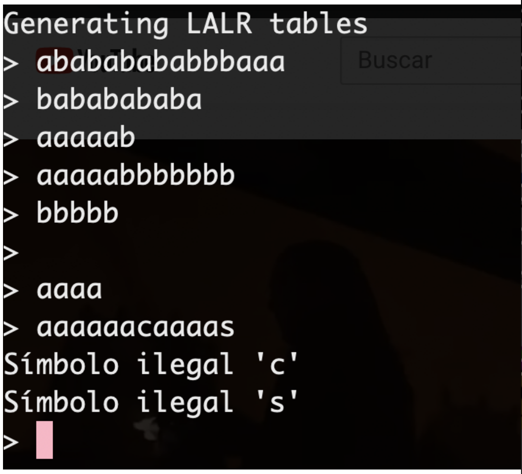
- Obtener una gramática regular para $(a^*b \cup b^*a)^*$ (por la izquierda).

    1. Realizamos exactamente los pasos (1-9) para la gramática anterior, pero en el paso 5 cambiamos

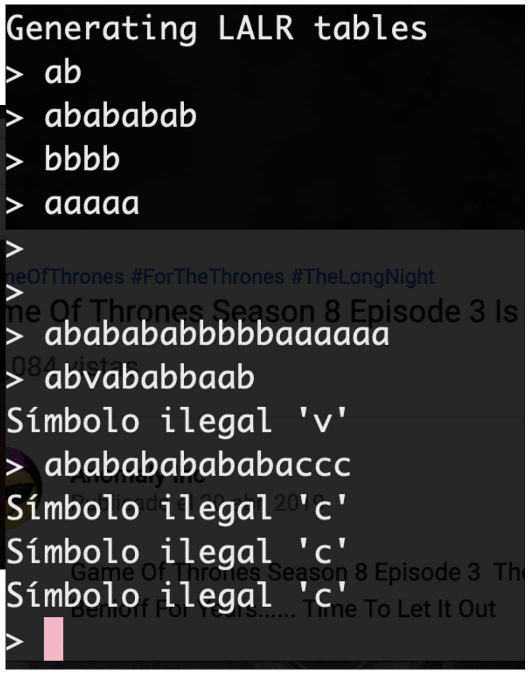
$$P : S \to aS|bS|\varepsilon \quad \text{a} \quad P : S \to Sa|Sb|\varepsilon$$

```
def p_S(p):
..''' S : a S
.......| b S
.......| empty '''
..pass
```

2. Ejecutamos el script e ingresamos cadenas fig 1 (b).



(a) Gramática (por la derecha).



(b) Gramática (por la izquierda).

Figura 1

3. Material y equipo.

- Computadora.
- Sistema operativo: Ubuntu 18.04.2 LTS.
- Editor de texto: Vim.
- Lenguaje: Python 3.

4. Conclusiones y recomendaciones.

El uso de gramáticas nos permite procesar lenguajes más complejos mediante un conjunto de reglas, las cuales permiten reconocer si ciertas cadenas son válidas para un lenguaje L. La implementación en python fue relativamente sencilla debido a la libreria `PLY`, la cual puede servir como base para construir herramientas más poderosas como compiladores.

Referencias

[1] Dean Kelley. (1995). Teoría de autómatas y lenguajes formales. Madrid: Pearson.

[2] Mark Lutz. (2013). Learning Python. Estados Unidos: O'Reilly.

[3] Python Software Foundation. (2019). Python 3.7.2 documentation. 20 Feb 2019, de Python Software Foundation Sitio web: https://docs.python.org/3/