



Instituto Politécnico Nacional
Escuela Superior de Cómputo

Teoría Computacional

Practica 6: Gramáticas Regulares

Alumna: Ramírez Galindo Karina

Profesor: Rosas Trigueros Jorge Luis

Fecha de realización: 04-11-2019

Fecha de entrega: 02-12-2019



Marco Teórico

🌈 **GRAMATICAS REGULARES**

Una *gramática regular* G es una 4-tupla. [1] [2]

$$G=(\Sigma, N, S, P)$$

Dónde:

- Σ es el alfabeto de entrada
- N es una colección de símbolos no terminales
- S es un no terminal llamado símbolo inicial
- P es una colección de reglas de sustitución llamadas producciones y que son de la forma

$$A \rightarrow w$$

Dónde: $A \in N$ y w es una cadena sobre $\Sigma \cup N$ que satisface

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 $L(G)$. [2]

➤ Ejemplo:

Considérese la gramática regular $G=(\Sigma,N,S,P)$, donde:

- $\Sigma = \{a, b\}$
- $N = \{S, A\}$
- $P : S \rightarrow bA$
- $A \rightarrow aaA \mid b \mid \varepsilon$

Obsérvese que $L(G)$ contendrá todas las cadenas de la forma:

$$ba^{2n}b \text{ y } ba^{2n}$$

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



De la definición se deduce que el lado derecho de una producción es una cadena de $\Sigma^*(NU\varepsilon)$. Obsérvese que ε puede ser el lado derecho de una producción. En el ejemplo precedente, la producción $A \rightarrow \varepsilon$ acaba con la generación de una cadena.

Dado que las producciones emparejan no terminales de N con cadenas de $\Sigma^*(NU\varepsilon)$, conviene representarlas como pares ordenados de $N \times \Sigma^*(NU\varepsilon)$. Por tanto, el par (x, y) de $N \times \Sigma^*(NU\varepsilon)$ representa a la producción $x \rightarrow y$. Las producciones de P del ejemplo anterior se podrían representar mediante

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

Si se llega al acuerdo de escribir los no terminales con letras mayúsculas y los terminales con letras minúsculas y además se conviene que S se use como símbolo inicial, entonces una gramática regular puede ser completamente especificada por medio de sus producciones. Por ejemplo, $S \rightarrow aS \mid b$ especifica completamente la gramática regular que genera el lenguaje a^*b .

Material y equipo

-  Sistema Operativo Ubuntu
-  Python 3

Desarrollo de la práctica

Objetivo: hacer un programa para procesar gramáticas regulares.

Ejercicios:

- 1) Realizar un programa en python que acepte la siguiente gramática regular.

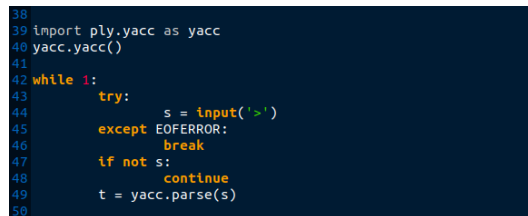
$S \rightarrow bA$

$A \rightarrow aaaA|b|\epsilon$



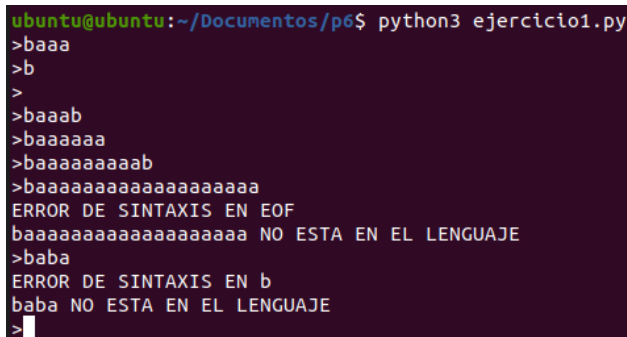
```
1 tokens = ('a','b');
2 t_a = r'a'
3 t_b = r'b'
4
5 def t_error(t):
6     print("CARACTER ILEGAL ", t.value[0])
7     t.lexer.skip(1)
8
9 import ply.lex as lex
10 lex.lex()
11
12 def p_S(p):
13     ''' S : b A'''
14     pass
15
16 def p_A(p):
17     ''' A : a a a A
18         | b
19         | empty '''
20
21     pass
22
23 def p_empty(p):
24     'empty :'
25     pass
26
27 s = ' '
28
29 def p_error(p):
30     global s
31     if p:
32         print("ERROR DE SINTAXIS EN", p.value)
33         print(s,"NO ESTA EN EL LENGUAJE")
34     else:
35         print("ERROR DE SINTAXIS EN EOF")
36         print(s,"NO ESTA EN EL LENGUAJE")
```

Imagen 1 Código del ejercicio 1 – Parte 1



```
38
39 import ply.yacc as yacc
40 yacc.yacc()
41
42 while 1:
43     try:
44         s = input('>')
45     except EOFError:
46         break
47     if not s:
48         continue
49     t = yacc.parse(s)
50
```

Imagen 2 Código del ejercicio 1 – Parte 2



```
ubuntu@ubuntu:~/Documentos/p6$ python3 ejercicio1.py
>baaa
>b
>
>baaab
>baaaaaa
>baaaaaaaaaab
>baaaaaaaaaaaaaaaaaaaaaa
ERROR DE SINTAXIS EN EOF
baaaaaaaaaaaaaaaaaaaaaa NO ESTA EN EL LENGUAJE
>baba
ERROR DE SINTAXIS EN b
baba NO ESTA EN EL LENGUAJE
>
```

Imagen 3

Compilación del ejercicio 1

Diseñar una gramática regular y probarla con PLY para:

2) $L_1 = \{ w \in \{a,b\}^* \mid w \text{ tiene un número impar de } a's \text{ y un número par de } b's \}$

Posibles cadenas:

$$w = \{\epsilon, abb, bab, bba, abbbb, \dots\}$$

La gramática queda de la siguiente forma:

$S \rightarrow abbA \mid bbaA \mid babA$

$A \rightarrow aaA \mid bbA \mid \epsilon$

```
Abrir  [F]
1 tokens = ('a','b');
2 t_a = r'a';
3 t_b = r'b';
4
5 def t_error(t):
6     print("CARACTER ILEGAL ", t.value[0])
7     t.lexer.skip(1)
8
9 import ply.lex as lex
10 lex.lex()
11
12 def p_S(p):
13     ''' S : a b b A
14         | b b a A
15         | b a b A
16         | empty '''
17     pass
18
19 def p_A(p):
20     ''' A : a a A
21         | b b A
22         | empty '''
23     pass
24
25 def p_empty(p):
26     'empty : '
27     pass
28
29 s = ''
30
31
32 def p_error(p):
33     global s
34     if p:
35         print("ERROR DE SINTAXIS EN", p.value)
36         print(s,"NO ESTA EN EL LENGUAJE")
37     else:
```

Imagen 4

Código del ejercicio 2 – Parte 1

```
37     else:
38         print("ERROR DE SINTAXIS EN EOF")
39         print(s,"NO ESTA EN EL LENGUAJE")
40
41
42 import ply.yacc as yacc
43 yacc.yacc()
44
45 while 1:
46     try:
47         s = input('>')
48     except EOFERROR:
49         break
50     if not s:
51         continue
52     t = yacc.parse(s)
```

Imagen 5

Código del ejercicio 2 – Parte 2

```
ubuntu@ubuntu:~/Documentos/p6$ python3 ejercicio2.py
>abb
>abbbb
>bab
>bababab
ERROR DE SINTAXIS EN b
bababab NO ESTA EN EL LENGUAJE
>baba
ERROR DE SINTAXIS EN EOF
baba NO ESTA EN EL LENGUAJE
>
```

Imagen 6

Compilación del ejercicio 2

$$3) L_2 = \{(a \cup c)^* \cup (a \cup c)^* b^+ \cup (a \cup c)^* b^+ a)^*\}$$

Posibles cadenas:

$$w = \{\epsilon, ac, acb, acbb, caac, cabb, \dots\}$$

La gramática queda de la siguiente forma:

$S \rightarrow A \mid \epsilon$

$A \rightarrow acA \mid caA \mid acAbB \mid caAbB \mid C \mid \epsilon$

$B \rightarrow Bb \mid \epsilon$

$C \rightarrow acAbBa \mid caAbBa \mid \epsilon$



```

1 tokens = ('a', 'b', 'c');
2 t_a = r'a';
3 t_b = r'b';
4 t_c = r'c';
5
6 def t_error(t):
7     print("CARACTER ILEGAL ", t.value[0])
8     t.lexer.skip(1)
9
10 import ply.lex as lex
11 lex.lex()
12
13 def p_S(p):
14     ''' S : A
15         | empty '''
16     pass
17
18 def p_A(p):
19     ''' A : a c A
20         | a c A b B
21         | c a A b B
22         | C
23         | empty '''
24
25     pass
26
27 def p_B(p):
28     ''' B : b B
29         | empty '''
30
31     pass
32
33 def p_C(p):
34     ''' C : a c A b B a
35         | c a A b B a
36         | empty '''
37

```

Imagen 7 Código del ejercicio 3 – Parte 1



```

38
39     pass
40
41 def p_empty(p):
42     'empty : '
43     pass
44
45 s = ' '
46
47 def p_error(p):
48     global s
49     if p:
50         print("ERROR DE SINTAXIS EN", p.value)
51         print(s, "NO ESTA EN EL LENGUAJE")
52     else:
53         print("ERROR DE SINTAXIS EN EOF")
54         print(s, "NO ESTA EN EL LENGUAJE")
55
56
57 import ply.yacc as yacc
58 yacc.yacc()
59
60 while 1:
61     try:
62         s = input('>')
63     except EOFError:
64         break
65     if not s:
66         continue
67     t = yacc.parse(s)

```

Imagen 8 Código del ejercicio 3 – Parte 2



```

Q ubuntu@ubuntu: ~/Documentos/p6
ubuntu@ubuntu:~/Documentos/p6$ python3 ejercicio3.py
Generating LALR tables
WARNING: 5 shift/reduce conflicts
WARNING: 4 reduce/reduce conflicts
WARNING: reduce/reduce conflict in state 3 resolved using rule (S -> empty)
WARNING: rejected rule (A -> empty) in state 3
WARNING: reduce/reduce conflict in state 3 resolved using rule (S -> empty)
WARNING: rejected rule (C -> empty) in state 3
WARNING: reduce/reduce conflict in state 10 resolved using rule (A -> empty)
WARNING: rejected rule (C -> empty) in state 10
WARNING: Rule (C -> empty) is never reduced
>ac
>acb
>acbb
>caac
>cabb
>cabbac
ERROR DE SINTAXIS EN c
cabbac NO ESTA EN EL LENGUAJE
>abc
ERROR DE SINTAXIS EN b
abc NO ESTA EN EL LENGUAJE
>cba
ERROR DE SINTAXIS EN b
cba NO ESTA EN EL LENGUAJE
>

```

Imagen 9 Compilación del ejercicio 3

Conclusiones

En esta práctica aprendí el uso ply.lex para programar gramáticas en python, es muy interesante este tema ya que las gramáticas generan un lenguaje y cada una tiene cierto nivel de complejidad pero cuando se logra obtenerla es fácil de entender.

Referencias

- [1] D. G. P. Luis Migel Pardo Vasallo, «Teoria de Automatas Finitos,» [En línea]. Available: https://ocw.unican.es/pluginfile.php/1516/course/section/1946/2-1_Introduccion.pdf.
- [2] J. E. Hopcroft, R. Motwani y J. D. Ullman, Introducción a la teoría de autómatas, lenguajes y computación, Madrid: PEARSON EDUCACIÓN S.A, 2007.