



UNIVERSIDAD
DE GRANADA

ESCUELA TECNICA SUPERIOR DE INGENIERIA
INFORMATICA Y TELECOMUNICACIONES

PRACTICAS MODELOS DE COMPUTACIÓN

Grupo B3

Juan Luis Torres Ramos

24 Octubre 2023

Practica 1

Encuentra una gramática libre del contexto para generar cada uno de los siguientes lenguajes:

1. $L = \{a^i b^j \mid i, j \in \mathbb{N}, i \leq j\}$.
2. $L = \{a^i b^j a^j b^i \mid i, j \in \mathbb{N}\}$.
3. $L = \{a^i b^i a^j b^j \mid i, j \in \mathbb{N}\}$.
4. $L = \{a_i b_i \mid i \in \mathbb{N}\} \cup \{b_i a_i \mid i \in \mathbb{N}\}$.
5. $L = \{uu^{-1} \mid u \in \{a, b\}^*\}$.
6. $L = \{a^i b^j c^{i+j} \mid i, j \in \mathbb{N}\}$.

donde \mathbb{N} es el conjunto de los numeros naturales incluyendo el 0

Pasos para resolver el ejercicio:

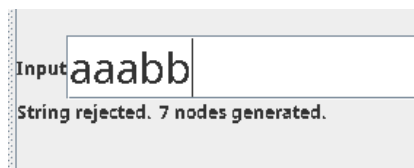
1. Determinar los símbolos terminales y no terminales.
2. Determinar el símbolo inicial.
3. Analizar el lenguaje para determinar qué se pide.
4. Determinar las reglas de producción.
5. Comprobar con JFLAP

A. $L = \{a^i b^j \mid i, j \in \mathbb{N}, i \leq j\}$.

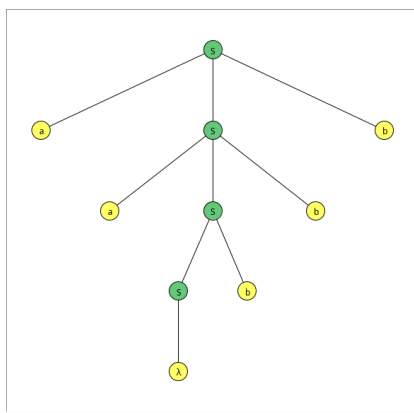
1. Los símbolos terminales serán $\{a, b\}$ y los símbolos no terminales serán S y B .
2. El símbolo inicial será S .
3. Analizar el lenguaje para determinar qué se pide. En este caso, se pide que la cadena tenga un número de a menor o igual que el número de b . Por ejemplo, $aabbb$ y $aabb$ pertenecen al lenguaje, pero aab no.
4. Determino las reglas de producción:
 - $S \rightarrow \epsilon$ (genero la cadena vacía).
 - $S \rightarrow aSb$.
 - $S \rightarrow Sb$.
5. compruebo con JFLAP que la gramática es correcta.

LHS		RHS
S	\rightarrow	λ
S	\rightarrow	aSb
S	\rightarrow	Sb

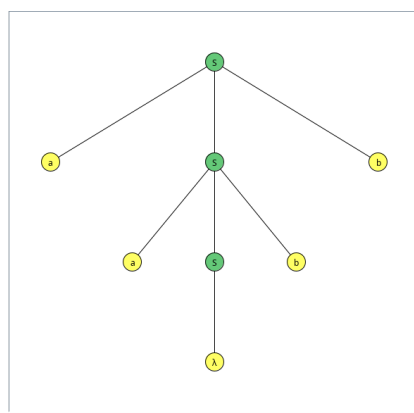
(a) la producción



(b) la cadena $aaabb$



(c) la cadena $aabbb$



(d) la cadena $aabb$

B. $L = \{a^i b^j a^j b^i \mid i, j \in \mathbb{N}\}$.

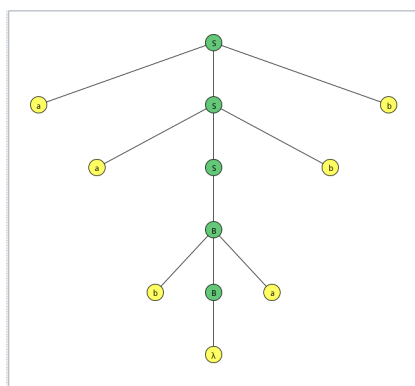
1. Los símbolos terminales serán $\{a, b\}$ y los símbolos no terminales serán S y B .
2. El símbolo inicial será S .
3. El lenguaje nos pide generar una cadena de 4 caracteres donde primero se generen $a^i b^j$ y luego $a^j b^i$, es decir en los extremos un número de caracteres i y en los caracteres del centro un número de caracteres j . Por ejemplo, $aababb$ y ab pertenecen al lenguaje, pero $aabbab$ no.
4. Determino las reglas de producción:
 - $S \rightarrow aSb$ (genero mismo número de caracteres en los extremos).
 - $S \rightarrow B$.
 - $B \rightarrow bBa$ (genero mismo número de caracteres en el centro).
 - $B \rightarrow \epsilon$ (genero la cadena vacía).
5. compruebo con JFLAP que la gramática es correcta.

LHS		RHS
S	\rightarrow	aSb
S	\rightarrow	B
B	\rightarrow	bBa
B	\rightarrow	λ

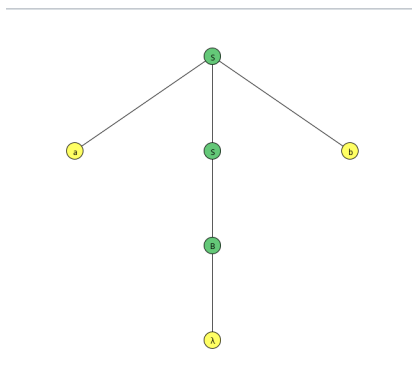
(a) la producción

Input: aabbab
String rejected. 5 nodes generated.

(b) la cadena $aabbab$



(c) la cadena $aababb$



(d) la cadena ab

C. $L = \{a^i b^i a^j b^j \mid i, j \in \mathbb{N}\}$.

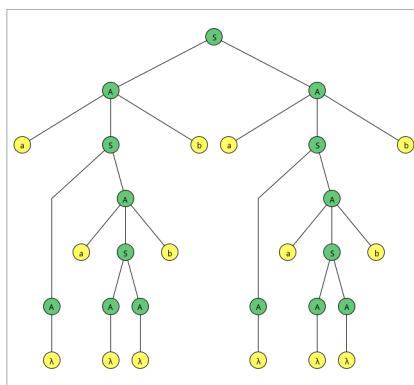
1. Los símbolos terminales serán $\{a, b\}$ y los símbolos no terminales serán S y A .
2. El símbolo inicial será S .
3. El lenguaje nos pide generar cadenas de 4 caracteres de la forma $abab$ donde los dos primeros caracteres tengan el mismo número de caracteres y para los dos últimos caracteres también tengan la misma cantidad. Ejemplos de cadenas serían $aabbaabb$, $aabbab$ pero no acepta $aaba$.
4. Determino las reglas de producción:
 - $S \rightarrow AA$ (símbolo inicial).
 - $A \rightarrow aSb$ (genero $\{a^i b^i \mid i \in \mathbb{N}\}$).
 - $A \rightarrow \epsilon$ (genero la cadena vacía).
5. compruebo con JFLAP que la gramática es correcta.

LHS		RHS
S	\rightarrow	AA
A	\rightarrow	aSb
A	\rightarrow	λ

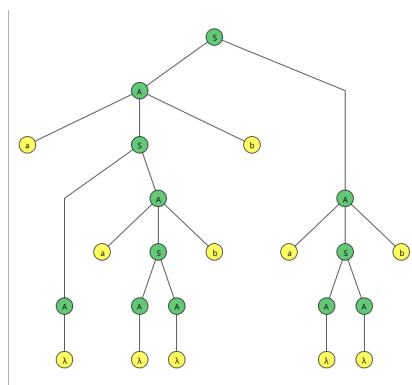
(a) la producción

Input: aaba
String rejected. 7 nodes generated.

(b) la cadena $aaba$



(c) la cadena $aabbaabb$



(d) la cadena $aabbab$

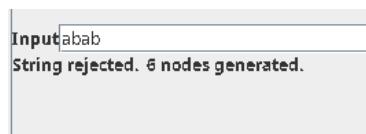
D. $L = \{a_i b_i \mid i \in \mathbb{N}\} \cup \{b_i a_i \mid i \in \mathbb{N}\}$.

1. Los símbolos terminales serán $\{a, b\}$ y los símbolos no terminales serán S, A, B .
2. El símbolo inicial será S .
3. Combina dos conjuntos de cadenas: el primero contiene cadenas de la forma $\{a_i b_i \mid i \in \mathbb{N}\}$, y el segundo contiene cadenas de la forma $\{b_i a_i \mid i \in \mathbb{N}\}$. Las cadenas $aabb, bbaa$ lo cumplen mientras $abab$ no lo cumple. Lo resolvemos por partes.
4. Determino las reglas de producción:
 - Podemos generar $\{a_i b_i \mid i \in \mathbb{N}\}$.
 $A \rightarrow aAb, A \rightarrow \epsilon$.
 - Por otro lado $\{b_i a_i \mid i \in \mathbb{N}\}$.
 $B \rightarrow bBa, B \rightarrow \epsilon$.
 - El lenguaje L se puede generar añadiendo.
 $S \rightarrow A, S \rightarrow B$.

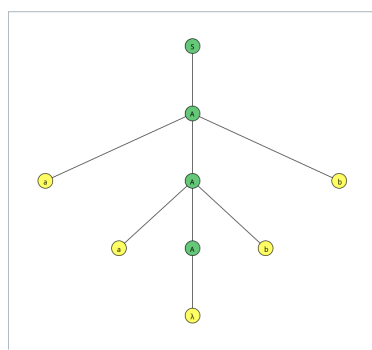
5. compruebo con JFLAP que la gramática es correcta.

LHS		RHS
S	\rightarrow	A
S	\rightarrow	B
A	\rightarrow	aAb
A	\rightarrow	λ
B	\rightarrow	bBa
B	\rightarrow	λ

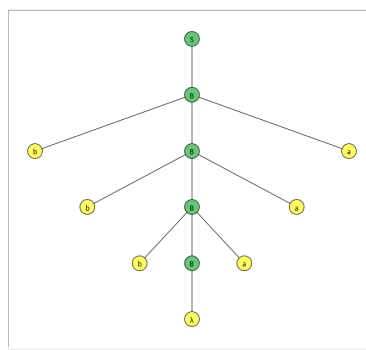
(a) la producción



(b) la cadena $abab$



(c) la cadena $aabb$



(d) la cadena $bbbaaa$

E. $L = \{uu^{-1} \mid u \in \{a, b\}^*\}$.

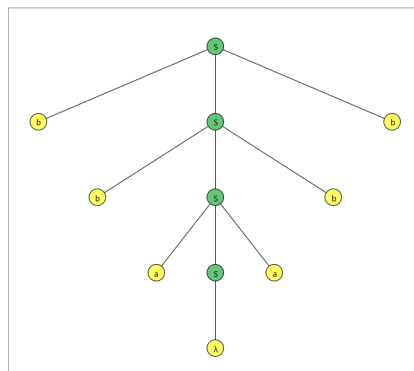
1. Los símbolos terminales serán $\{a, b\}$ y los símbolos no terminales serán S .
2. El símbolo inicial será S .
3. Analizar el lenguaje para determinar qué se pide. En este caso, se pide generar cadenas que son palíndromos formados por caracteres 'a' y 'b'. Cadenas que pertenecen al lenguaje son *abba* y *bbaabb* pero no *bbabb*.
4. Determino las reglas de producción:
 - $S \rightarrow \epsilon$ (genero la cadena vacía).
 - $S \rightarrow aSa$.
 - $S \rightarrow bSb$.
5. compruebo con JFLAP que la gramática es correcta.

LHS		RHS
S	\rightarrow	aSa
S	\rightarrow	bSb
S	\rightarrow	λ

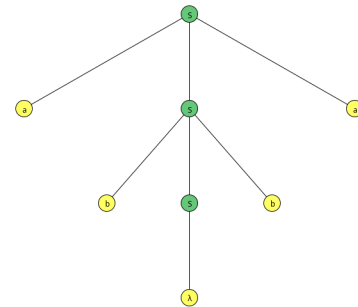
(a) la producción

Input:
String rejected. 5 nodes generated.

(b) la cadena *bbab*



(c) la cadena *bbaabb*



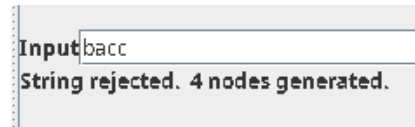
(d) la cadena *abba*

F. $L = \{a^i b^j c^{i+j} \mid i, j \in \mathbb{N}\}$.

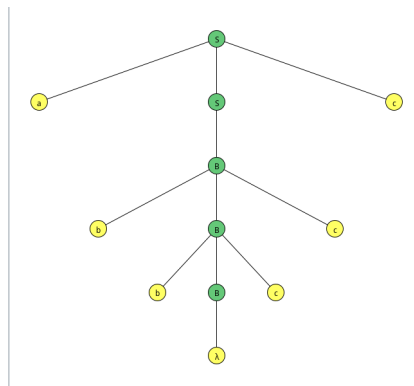
1. Los símbolos terminales serán $\{a, b, c\}$ y los símbolos no terminales serán S .
2. El símbolo inicial será S .
3. En este caso, se pide generar cadenas donde la cantidad de 'a's y 'b's es igual y la cantidad total de 'c's es la suma de las cantidades de 'a' y 'b'. Cadenas que cumplen la gramática son *abbccc* y *aaabccccc* pero no *bacc*.
4. Determino las reglas de producción:
 - $S \rightarrow aSc$ (genero la cadena vacía).
 - $S \rightarrow B$.
 - $B \rightarrow bBc$.
 - $B \rightarrow \epsilon$.
5. compruebo con JFLAP que la gramática es correcta.

LHS		RHS
S	\rightarrow	aSc
S	\rightarrow	B
B	\rightarrow	bBc
B	\rightarrow	λ

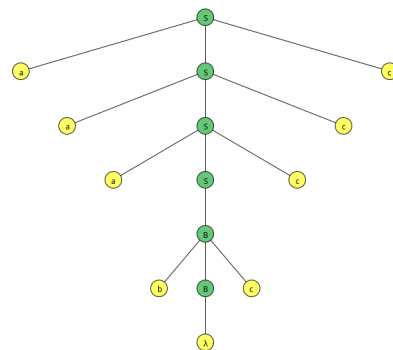
(a) la producción



(b) la cadena *bacc*



(c) la cadena *abbccc*



(d) la cadena *aaabccccc*

Practica 2

Analizadores léxicos, problemas de minería, trabajo Lex, 2 problema

Tareas a realizar

1. Formar un grupo de trabajo compuesto por una, dos o tres personas.
2. Cada grupo de trabajo debe pensar un problema original de procesamiento de textos. Para la resolución de este problema debe ser apropiado el uso de Lex, o sea, se debe resolver mediante el emparejamiento de cadenas con expresiones regulares y la asociación de acciones a cada emparejamiento.
3. Cada grupo debe resolver el problema propuesto usando Lex. Se deberá realizar una memoria donde se presente una descripción del problema y su solución, además de entregar electrónicamente los ficheros de texto con la implementación de la solución.
4. Esta práctica deberá ser entregada antes del día 31 de Diciembre de 2020. Se entregará a través de la plataforma PRADO en un fichero .zip conteniendo todos los archivos de esta práctica. Sólo es necesario que lo entregue uno de los componentes del grupo.

Pasos para resolver el ejercicio:

1. Descripción del problema
2. solución
3. código lex

1. Descripción del Problema

Soy un nuevo profesor de la asignatura de Fundamentos de Programación. Tras corregir varios ejercicios de los alumnos me he dado cuenta que la cantidad de comentarios explicando el código va relacionada con la nota del ejercicio. Por lo que he decidido crear un programa que calcule la densidad de comentarios en un código fuente en C para evaluar positivamente a los alumnos que comenten su código.

Densidad de comentarios código

Tu tarea es desarrollar un programa en Lex que calcule la densidad de comentarios en un código fuente en C. La densidad de comentarios se define como el porcentaje del código total que está ocupado por comentarios.

Pasos

El alumno ha entregado su ejercicio de C correspondiente de la asignatura, voy a calcular la densidad de comentarios con la siguiente fórmula:

$$\text{Densidad de comentarios} = \frac{\text{Total de letras en un comentario}}{\text{Total de letras en el código}}$$

1. Creo 2 variables globales, para contar letras en el código y en comentarios.
2. Defino 2 estados `INCOMMENTBLOCK` y `INCOMMENTLINE` para manejar por separado los dos tipos de comentarios en C: comentarios en línea y comentarios en bloque.
3. Defino una función `contarLetras` que cuenta únicamente letras; no cuenta espacios en blanco, tabuladores, saltos de línea ni retornos de carro.
4. Defino reglas de Flex para reconocer los comentarios:
 - Si encuentra `/*`, comienza el subestado `INCOMMENTBLOCK` y termina con `*/`.
 - Si encuentra `/**`, comienza el subestado `INCOMMENTLINE` y termina con un salto de línea.
 - Dentro del estado `INCOMMENTBLOCK`, selecciono para cualquier carácter que no sea un asterisco (para evitar contar el fin del comentario `*/`) ni un salto de línea. Para comentarios en línea, solo no cuento el salto de línea.
 - El texto seleccionado corresponde a la variable `yytext`, la cual introduzco en mi función `contarLetras`.
 - Imprimo cada comentario encontrado haciendo `print` a `yytext`, también indico tipo comentario y su longitud.
 - Para referirme a todo el código, uso `.\n`, haciendo referencia a cualquier carácter y un salto de línea.

5. Por último, calculo la densidad de comentarios con la fórmula anterior.

Para ejecutar el programa usaremos un Makefile

```
1 $ make all
2 $ make run
```

Listing 1: Ejemplo de ejecución

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  // funcion de ejemplo
5  void imprimirMensaje() {
6      printf("! Hola, mundo!\n");
7  }
8
9  /*
10     comentarios
11     multilinea
12 */
13
14  int main{
15      // Llamada a la funcion
16      imprimirMensaje();
17      return 0;
18  }
19
```

Listing 2: Ejemplo de Entrada

```
./comentador < ./ejemplos/ejemplo1.c
-----
DEBUG:

Comentario en línea:  funcion de ejemplo
Número de letras en comentario en línea: 16

Comentario en bloque:  comentarios      multilinea
Número de letras en comentario en bloque: 21

Comentario en línea:  Llamada a la funcion
Número de letras en comentario en línea: 17

-----
RESULTADO:

Número total de letras en código: 118
Número total de letras en comentarios: 54
Porcentaje de Comentarios: 45.76 %
```

Figure 7: Resultado de la ejecución del programa

Analisis Resultado

Podemos ver que el programa ha detectado los comentarios y diferenciado si es un comentario en línea o un comentario en bloque. Calcula correctamente tanto las letras de cada comentario como el total de letras en el código y por último calcula el porcentaje de comentarios en el código correspondiente. Ahora el maestro, viendo el porcentaje de comentarios de cada ejercicio de FP, puede evaluar positivamente a los alumnos que comenten su código correctamente. Tampoco se puede abusar de los comentarios, ya que el maestro puede ver el porcentaje de comentarios y si es demasiado alto puede penalizar al alumno.

```

1  %{
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  int total_letters = 0;
6  int comment_letters = 0;
7  int comment_letters_global = 0;
8  int contarLetras(const char* texto);
9  %}
10
11 %x INCOMMENTBLOCK INCOMMENTLINE
12
13 %%
14 "/*" {
15     BEGIN(INCOMMENTBLOCK);
16     printf("\nComentario en bloque: ");
17     comment_letters = 0;
18 }
19
20 <INCOMMENTBLOCK>"*/" {
21     BEGIN(INCOMMENTBLOCK);
22     printf("\nNumero de letras en comentario en bloque: %d\n",
23 comment_letters);
24     comment_letters = 0;
25 }
26
27 <INCOMMENTBLOCK>[^*\n]+ {
28     int letras_comentario = contarLetras(yytext);
29     printf("%s", yytext);
30     comment_letters += letras_comentario;
31     comment_letters_global += letras_comentario;
32 }
33
34 <INCOMMENTBLOCK>\n {
35     // No contar el salto de linea
36 }
37
38 "/*" {
39     BEGIN(INCOMMENTBLOCK);
40     printf("\nComentario en bloque: ");
41 }
42
43 <INCOMMENTLINE>\n {
44     BEGIN(INCOMMENTLINE);
45     printf("\nNumero de letras en comentario en linea: %d\n",
46 comment_letters);
47     comment_letters = 0;
48 }
49
50 <INCOMMENTLINE>[^*\n]+ {
51     printf("%s", yytext);
52     comment_letters += contarLetras(yytext);
53     comment_letters_global += contarLetras(yytext);
54 }
55
56 .|\n {
57     total_letters += contarLetras(yytext);

```

```

56     }
57     %%
58     // sin salto de linea al final, solo cuento letras simbolos y
    numeros
59     int contarLetras(const char* texto) {
60         int contador = 0;
61         while (*texto) {
62             if (*texto != '\n' && *texto != '\t' && *texto != ' '
&& *texto != '\r') {
63                 contador++;
64             }
65             texto++;
66         }
67         return contador;
68     }
69     // conslato de linea
70     int main(int argc, char* argv[]) {
71         printf("DEBUG:\n");
72         yylex();
73
74         printf("RESULTADO:\n");
75         printf("\nNumero total de letras en codigo: %d\n",
total_letters);
76         printf("Numero total de letras en comentarios: %d\n",
comment_letters_global );
77         printf("Porcentaje de Comentarios: %.2f\n", (float)(
comment_letters_global) / total_letters * 100);
78         return EXIT_SUCCESS;
79     }
80

```

Listing 3:Codigo Analizador Lex

```

1      all: comentador
2
3      comentador: lex.yy.c
4                  gcc -o comentador lex.yy.c -lfl
5
6      lex.yy.c: comentador.l
7                  flex comentador.l
8
9      run: comentador
10         ./comentador < ./ejemplos/ejemplo1.c
11         ./comentador < ./ejemplos/ejemplo2.c
12

```

Listing 4: Makefile

Practica 3

Diseña una máquinas de estados finitos, en particular la máquina de Mealy, para simular la codificación y decodificación del código Enigma. Implementa un conjunto de estados y transiciones que reflejen el proceso de cifrado y descifrado característico del Enigma. Utiliza JFLAP para simular y visualizar la máquina de Mealy que actúa como codificador y decodificador.

El código enigma es una máquina de cifrado mecánica utilizada durante la 2ª guerra mundial. Funciona mediante rotores giratorios que cifran las letras del alfabeto

Funcionamiento

1. Los rotores tenían una posición inicial
2. Cuando se presiona una tecla, la corriente pasa por los rotores y cifra la letra con la correspondiente con el rotor
3. El rotor (o rotores) gira después de cifrar una letra
4. la corriente eléctrica, que tiene la letra, pasa por un reflector, que refleja la señal de vuelta a través de los rotores
5. al volver a pasar por los rotores se vuelve a cifrar
6. se obtiene el código

Pasos a seguir para crear máquina de Mealy

1. Definir los Estados de la máquina de Mealy: Inicio, rotor,
2. Definir alfabeto de entrada A,E,I,O,U
3. Diseñar las transiciones entre estados (movimiento rotores, configuración rotores)