



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:

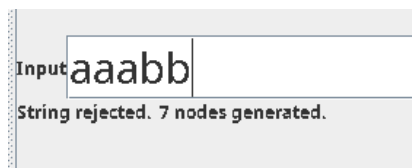
1. Determinar los símbolos terminales y no terminales.
2. Determinar el símbolo inicial.
3. Analizar el lenguaje para determinar qué se pide.
4. Determinar las reglas de producción.
5. Comprobar con JFLAP

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

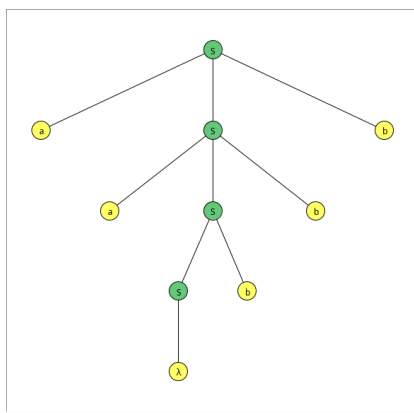
1. Los símbolos terminales serán $\{a, b\}$ y los símbolos no terminales serán S y B .
2. El símbolo inicial será S .
3. Analizar el lenguaje para determinar qué se pide. En este caso, se pide que la cadena tenga un número de a menor o igual que el número de b . Por ejemplo, $aabbb$ y $aabb$ pertenecen 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

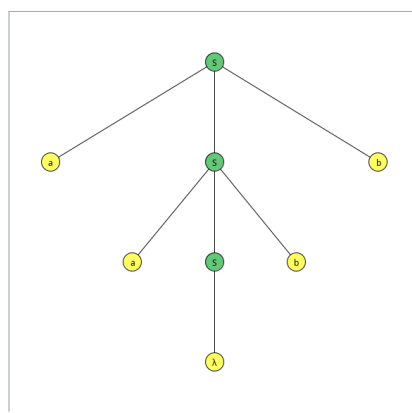
(a) la producción



(b) la cadena $aaabb$



(c) la cadena $aabbb$



(d) la cadena $aabb$

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

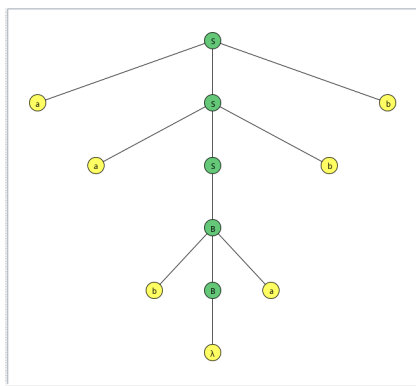
1. Los símbolos terminales serán $\{a, b\}$ y los símbolos no terminales serán S y B .
2. El símbolo inicial será S .
3. El lenguaje nos pide generar una cadena de 4 caracteres donde primero se generen $a^i b^j$ y luego $a^j b^i$, es decir en los extremos un número de caracteres i y en los caracteres 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

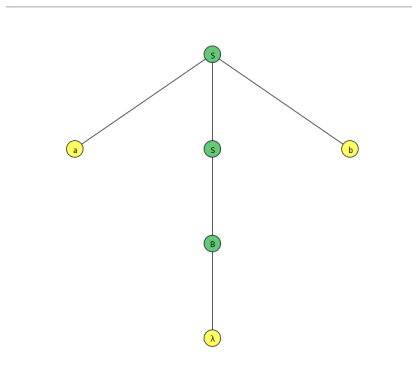
(a) la producción

Input: aabbab
String rejected. 5 nodes generated.

(b) la cadena $aabbab$



(c) la cadena $aababb$



(d) la cadena ab

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

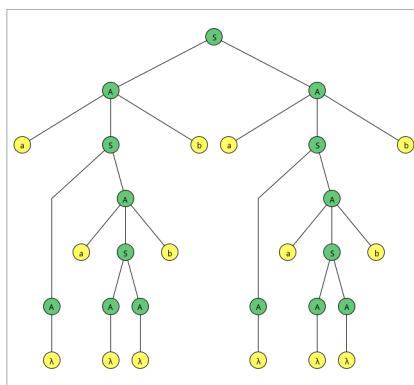
1. Los símbolos terminales serán $\{a, b\}$ y los símbolos no terminales serán S y A .
2. El símbolo inicial será S .
3. El lenguaje nos pide generar cadenas de 4 caracteres de la forma $abab$ donde los dos primeros caracteres tengan el mismo número de caracteres y para los dos últimos caracteres también 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

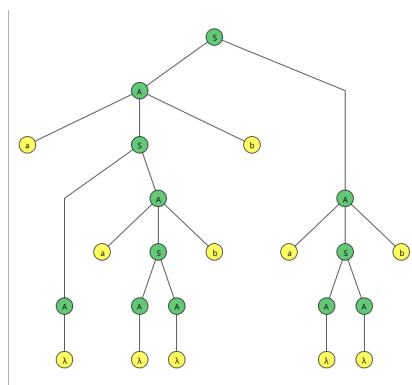
(a) la producción

Input: aaba
String rejected. 7 nodes generated.

(b) la cadena $aaba$



(c) la cadena $aabbaabb$



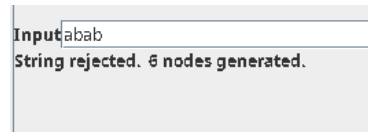
(d) la cadena $aabbab$

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

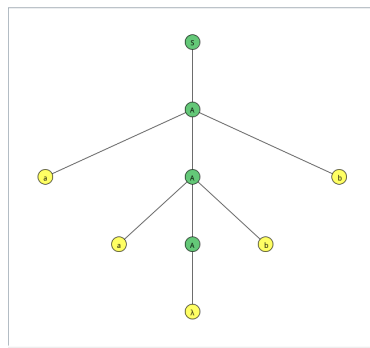
1. Los símbolos terminales serán $\{a, b\}$ y los símbolos no terminales serán S, A, B .
2. El símbolo inicial será S .
3. Combina dos conjuntos de cadenas: el primero contiene cadenas de la forma $\{a_i b_i \mid i \in \mathbb{N}\}$, y el segundo contiene cadenas de la forma $\{b_i a_i \mid i \in \mathbb{N}\}$. 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

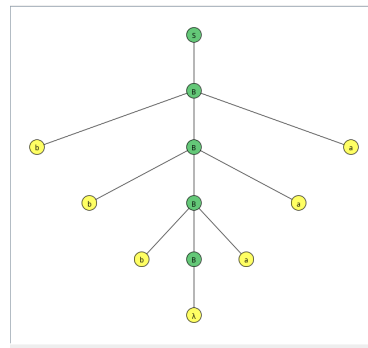
(a) la producción



(b) la cadena $abab$



(c) la cadena $aabb$



(d) la cadena $bbbaaa$

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

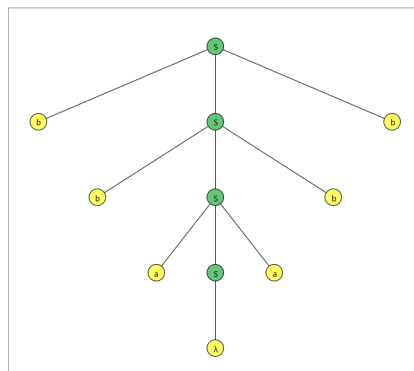
1. Los símbolos terminales serán $\{a, b\}$ y los símbolos no terminales serán S .
2. El símbolo inicial será S .
3. Analizar el lenguaje para determinar qué se pide. En este caso, se pide generar cadenas que son palíndromos formados por caracteres 'a' y 'b'. Cadenas que pertenecen al lenguaje son *abba* 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

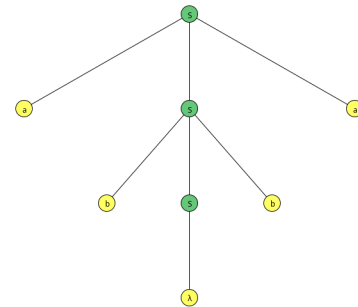
(a) la producción

Input:
 String rejected. 5 nodes generated.

(b) la cadena *bbab*



(c) la cadena *bbaabb*



(d) la cadena *abba*

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

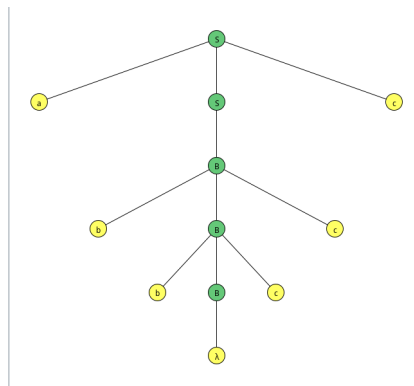
1. Los símbolos terminales serán $\{a, b, c\}$ y los símbolos no terminales serán S .
2. El símbolo inicial será S .
3. En este caso, se pide generar cadenas donde la cantidad de 'a's y 'b's es igual y la cantidad total de 'c's es la suma de las cantidades de 'a' y 'b'. Cadenas que cumplen la gramática son $abbc$ y $aaabcccc$ 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

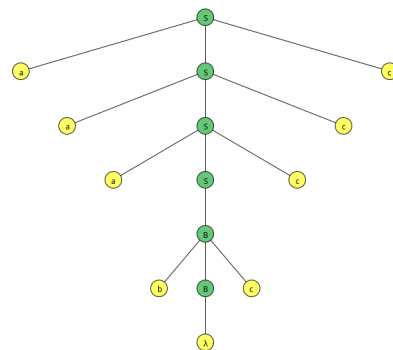
(a) la producción

Input: bacc
String rejected. 4 nodes generated.

(b) la cadena *bacc*



(c) la cadena *abbccc*



(d) la cadena *aaabcccc*

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.

Ejemplo

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

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

para resolver el problema tendré que reconocer los comentarios. En C se usa `"/ * */` y `"/ /"`, luego calcular el total de letras de un comentario y el total de letras en el código, por último usaremos la función anterior

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`, para cualquier carácter que no sea un asterisco (para evitar contar el fin del comentario `*/`) ni un salto de línea (`\n`). 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, indicando su tipo 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.

Entrada

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  // funcion de ejemplo
6  void imprimirMensaje() {
7      printf("¡ Hola, mundo!\n");
8  }
9
10 /*
11  comentarios
12  multilinea
13  */
14
15 int main{
16     // Llamada a la funcion
17     imprimirMensaje();
18     return 0;
19 }
20
```

Practica 3

máquinas de estados para codificar y decodificar código Enigma. Usando las máquinas de estado finito, en concreto la de Mealy, hay que implementar un código para con codificador y después se decodifica. El codificador y el decodificador, se simula en JFlag como una máquina de Mealy