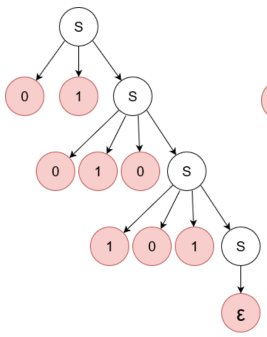
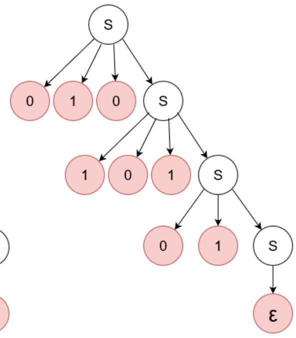
Práctica 4: Ejercicios prácticos sobre Lenguajes libres de contexto

1) Determinar cuáles de las siguientes gramáticas son ambiguas y, en su caso, comprobar si los lenguajes generados son inherentemente ambiguos. Justificar la respuesta.

a) S -> 01S | 010S | 101S | ε

Esta gramática es ambigua, ya que para obtener la palabra 01010101 se pueden desarrollar dos árboles de derivación al menos.

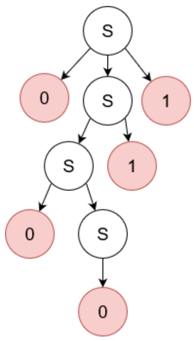


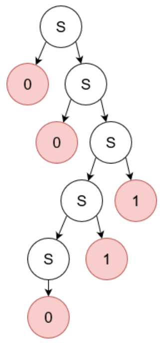


Por su parte el lenguaje que nos da esta gramática es un lenguaje que puede empezar por 01 o 10, luego tener cualquier combinación de ceros y unos mientras no haya más de dos ceros o dos unos juntos, y tiene que terminar en 01 o 10. Todas las gramáticas que he encontrado para este lenguaje son ambiguas, por lo tanto, asumo que el lenguaje es inherentemente ambiguo.

b) S -> 0S1 | S1 | 0S | 0

Esta gramática es ambigua porque por ejemplo para la palabra 00011 hay al menos dos árboles de derivación.





El lenguaje que genera esta gramática es $L(G) = \{0^n1^m \ / \ n \geq 1 \ m \geq 0\}$.

Pero para este lenguaje si he encontrado otra gramática que lo genera la cual no es ambigua, por lo tanto, el lenguaje no es inherentemente ambiguo. La gramática que he encontrado es la siguiente:

$S \to 0S \mid 0S_1$ $\quad$ $S_1 \to 1S_1 \mid \varepsilon$

c) S -> A1B $\qquad$ **A -> 0A | ε** $\qquad$ **B -> 0B | 1B | ε**

Esta gramática no es ambigua, ya que no he encontrado ninguna palabra que pueda tener dos árboles de derivación distintos, además de que el lenguaje que genera es $L(G) = \{0^n1u \ / \ n \geq 0 \ \ u \in A^*\}$.

Como vemos todas las palabras que puede generar están muy bien cuadradas, es decir, siempre va una secuencia de ceros (puede ser ninguno), los cuales solo se pueden poner con las reglas de derivación de la variable A y siempre se añaden a la izquierda de la palabra. Luego tiene que ir obligatoriamente un 1, y por último, después del 1 puede ir absolutamente cualquier combinación de ceros y unos (puede no ir ningún 0 ni ningún 1), pero esta combinación se genera solamente con las reglas de derivación de la variable B que como vemos ponen un símbolo terminar y luego otra vez la variable B de modo que las reglas siempre se van a ir disparando en el orden en que el aparezcan los símbolos en la palabra, sin poder dar lugar a que haya más de un árbol de derivación posible por palabra. Por lo tanto, podemos afirmar que es una gramática no ambigua.

2) Eliminar símbolos y producciones inútiles. Realizar el procedimiento paso por paso, indicando las variables descartadas y el motivo.

S -> moA;	**S -> cI;**	**A -> dEs;**	**A -> jBI;**
B -> bb;	**B -> D;**	**E -> elO;**	**E -> Perl;**
D -> de;	**C -> c;**	**J -> kC;**	**I -> fI;**
O -> o;	**P -> oIa;**		

Lo primero que hacemos es identificar las reglas que derivan únicamente en símbolos terminales, meter en V_T la variable a la izquierda de la regla y "quitar" todas las reglas con esa variable a la izquierda de lista para no tenerlas en cuenta las siguientes pasadas. V_T queda de la siguiente manera:

$V_T = \{B, C, D, O\}$

El siguiente paso es volver a repasar la lista de reglas que nos quedan para ver si en su parte derecha todas las variables que tienen están en V_T, y en caso de que así sea, meter la variable a la izquierda de la regla en V_T y "quitar" todas las reglas con esa variable a la izquierda de la lista. Tras la segunda pasada a la lista V_T quedaría así:

$V_T = \{B, C, D, O, E, J\}$

Ahora volvemos a repetir el paso anterior, y seguiremos haciéndolo hasta que al recorrer la lista de reglas ninguna se añada a V_T.

3ª pasada -> V_T = {B, C, D, O, E, J, A}
4º pasada -> V_T = {B, C, D, O, E, J, A, S}

En la quinta pasada ya no se añade ninguna variable a V_T, por lo que ahora lo que tenemos que hacer es eliminar las reglas que tengas las variables que no se han metido en V_T, es decir {I, P}, y por lo tanto las reglas que nos quedan son:

S -> moA;		A -> dEs;
B -> bb;	B -> D;	E -> elO;
D -> de;	C -> c;	J -> kC;
O -> o;		

Ahora vamos analizando las reglas que nos quedan una a una, y metiendo en V_s las variables analizadas, en J las variables a analizar, y en T_s los símbolos terminales.

Para empezar, tenemos J = {S} para empezar, así que analizamos su regla y metemos en V_s = {S}, J = {A} y en T_s = {m, o}.

Seguimos con la siguiente variable en J, la cual es A y la analizamos de modo que nos queda en V_s = {S, A}, J = {E} y en T_s = {m, o, d, s}.

Continuamos con la E y ahora tenemos en V_s = {S, A, E}, J = {O} y en T_s = {m, o, d, s, e, l}.

Y finalmente analizamos la última variable en J, la cual es la O, y se nos queda en V_s = {S, A, E, O}, J = {} y en T_s = {m, o, d, s, e, l}.

Por lo que las reglas que no tengan a su lado izquierdo una de las variables de V_s significa que son inútiles ya que no son accesibles, por lo que la gramática finalmente se nos quedaría de la siguiente forma:

S -> moA; A -> dEs; E -> elO; O -> o;

Que como vemos nos da el siguiente resultado:

S => moA => modEs => modelOs => modelos

3) <u>Eliminar producciones nulas y unitarias, en el orden correcto. Realizar los procedimientos paso por paso, indicando las producciones descartadas en cada momento.</u>

S -> XYZ	**S -> XYz**	**X -> xxX**	**X-> ε**
Y -> yyY	**Y -> ε**	**Z -> yxZ**	**Z -> X**

Lo primero que tenemos que hacer es meter en H el conjunto de variables anulables, las cuales son las que derivan en la cadena vacía, por lo tanto, para empezar, metemos en H = {X, Y}, y acto seguido metemos también Z por la regla de Z -> X, y como hemos metido Z, tenemos que meter S por la regla de S -> XYZ, de modo que H = {X, Y, Z, S}.

Lo siguiente es quitar las producciones que dan a la cadena vacía directamente, con lo que la gramática se nos queda de la siguiente forma:

S -> XYZ	S -> XYz	X -> xxX
Y -> yyY	Z -> yxZ	Z -> X

Ahora tenemos que para cada regla que nos queda crear otras a partir de ellas, probando a quitar las distintas combinaciones posibles con las variables en H, así de S -> XYZ saldrían

S -> XY   S -> YZ   S -> XZ   S-> X   S -> Y   S -> Z

De S -> XYz salen: S -> Yz   S -> Xz   S -> z

De X -> xxX sale: X -> xx

De Y -> yyY sale: Y -> yy

Y por último de Z -> yxZ sale: Z-> yx

Así la gramática que tenemos ahora mismo es:

S -> XYZ	S -> XYz	X -> xxX	Y -> yyY	Z -> yxZ	Z -> X
S -> XY	S -> YZ	S -> XZ	S-> X	S -> Y	S -> Z
S -> Yz	S -> Xz	S -> z	X -> xx	Y -> yy	Z-> yx

Y ahora toca eliminar las producciones unitarias, para ello metemos en H las reglas que solo deriven a una variable, y metemos los dos lados de la producción. Quedando de la siguiente manera H = {(Z, X), (S, X), (S, Y), (S, Z)}

Y eliminamos dichas producciones de la gramática, pero para compensarlo tenemos que poner las distintas combinaciones que sale de dejar en el lado izquierdo la variable que estaba, y en el lado derecho poner las distintas derivaciones que se nos dan desde la variable que está en ese lado.

Así de Z -> X, al eliminarla, tenemos que añadir Z -> xxX   Z -> xx

De S -> X tenemos que añadir S -> xxX   S -> xx

De S -> Y tenemos que añadir S-> yyY   S-> yy

Y de S -> Z tenemos que añadir S -> yxZ   S -> yx

Por lo tanto, la gramática final resultante es:

S -> XYZ	S -> XYz	X -> xxX	Y -> yyY	Z -> yxZ	S -> XY
S -> YZ	S -> XZ	S -> Yz	S -> Xz	S -> z	X -> xx
Y -> yy	Z-> yx	Z -> xxX	Z -> xx	S -> xxX	S -> xx
S-> yyY	S-> yy	S -> yxZ	S -> yx		

4) <u>Pasar la siguiente gramática a forma normal de Greibach:</u>

S -> a |CD | CS

A -> a | b | SS

C -> a

D -> AS

Si analizamos la gramática que nos dan, vemos que está en forma normal de Chomsky, por lo tanto, para pasarla a la forma normal de Greibach solo hay que hacer el último paso, el cual es cambiar las producciones para que todas sean del tipo A -> aα, donde A es una variable, a un símbolo terminal, y α es un conjunto de variables (puede ser ninguna).

Comenzamos por la última producción, es decir, D -> AS, y lo que hacemos es eliminarla, sustituyendo A por sus posibles derivaciones. De esta manera nos aparecen:

D -> SSS   D -> aS   D -> bS

De las cuales solo D -> SSS no está en la forma que buscamos, así que repetimos el proceso sobre ella, es decir, la eliminamos y obtenemos las producciones correspondientes a las derivaciones de la primera S.

D -> aSS   D-> CDSS   D -> CSSS

Ahora son D -> CDSS y D -> CSSS sobre las cuales necesitamos hacer el proceso, y así obtenemos:

D -> aDSS D -> aSSS

Como ya todas las que tienen a la izquierda D están como queremos, pasamos a las que tiene la A a la izquierda, y vemos que A -> SS no está como nos interesa, así que aplicamos el procedimiento anterior y obtenemos:

A -> aS A -> CDS A ->CSS

Nuevamente, nos salen dos que no nos interesan, sobre las cuales aplicar el procedimiento para obtener:

A -> aDS A -> aSS

Y ya por último nos quedan las que tiene la S a la izquierda, de las cuales son S -> CD y S -> CS las que no nos interesan y las eliminamos obteniendo a cambio:

S -> aD S -> aS

Así que al final la gramática resultante en forma normal de Greibach es:

S -> a	S -> aD	S -> aS	A -> a	A -> b
A -> aS	A -> aDS	A -> aSS	C -> a	D -> aS
D -> bS	D -> aSS	D->aDSS	D -> aSSS	