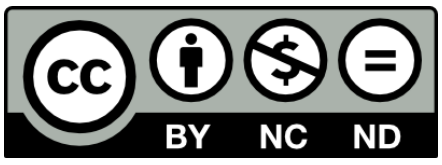
Modelos de Computación

Los Del DGIIM, losdeldgiim.github.io

Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada



Modelos de Computación

Los Del DGIIM, losdeldgiim.github.io

Arturo Olivares Martos

Granada, 2024-2025

Índice general

1. Relaciones de Problemas

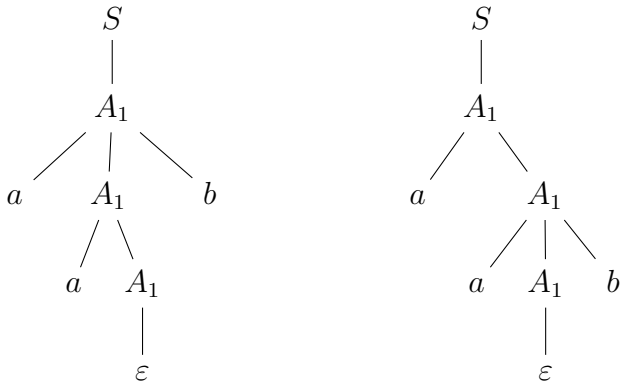


Figura 1.1: Árboles de derivación para *aab* usando la Gramática del Ejercicio 1.1.1.

1.1. Gramáticas Independientes del Contexto

Observación. Salvo que se indique lo contrario, las letras en mayúsculas representan variables, las letras en minúsculas representan terminales y la S representa el símbolo inicial.

Ejercicio 1.1.1. Determinar si la siguiente gramática es ambigua y si el lenguaje generado es inherentemente ambiguo:

$$\begin{cases} S \to A_1 \mid A_2 \\ A_1 \to aA_1b \mid aA_1 \mid \varepsilon \\ A_2 \to aA_2b \mid A_2b \mid \varepsilon \end{cases}$$

La gramática dada es ambigua puesto que hay palabras con más de un árbol de derivación. Por ejemplo, la palabra *aab* tiene los dos posibles árboles de derivación que se muestran en la Figura 1.1.

Veamos ahora que no es inherentemente ambiguo. La producción de A_1 produce las palabras de la forma a^ib^j, con $i \geqslant j$. La producción de A_2 produce las palabras de la forma a^ib^j, con $i \leqslant j$. Por tanto, la gramática genera el lenguaje:

$$\begin{aligned} L &= \{a^ib^j \mid i,j \in \mathbb{N} \cup \{0\},\ i \geqslant j\} \cup \{a^ib^j \mid i,j \in \mathbb{N} \cup \{0\},\ i \leqslant j\} = \\ &= \{a^ib^j \mid i,j \in \mathbb{N} \cup \{0\}\} \end{aligned}$$

Este lenguaje es regular con expresión regular asociada:

$$a^*b^*$$

Por tanto, como es regular, tenemos que no es inherentemente ambiguo.

Ejercicio 1.1.2. Sea la gramática

$$\begin{cases} S \to aSA \mid \varepsilon \\ A \to bA \mid \varepsilon \end{cases}$$

1. Demostrar que es ambigua.

   Tomemos como palabra *aabb*. Esta palabra tiene dos árboles de derivación, como se muestra en la Figura 1.2.

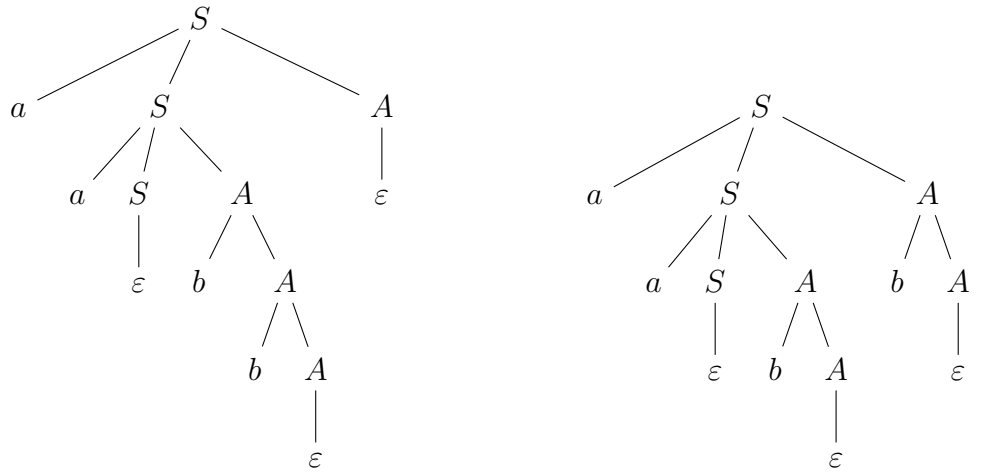


Figura 1.2: Árboles de derivación para *aabb* usando la Gramática del Ejercicio 1.1.2.

2. Dar una expresión regular para el lenguaje generado.

   En primer lugar, hemos de considerar que $\varepsilon \in L$. Además, todas las palabras de longitud positiva empiezan por a. por tanto, la expresión regular para el lenguaje generado por la gramática es:

$$a^+b^* + \varepsilon$$

3. Construir una gramática no ambigua que genere el mismo lenguaje.

   Una primera opción sería obtener el autómata y pasar este a gramática. No obstante, consideramos directamente la gramática $G = (V, \{a, b\}, P, S)$, con:

$$V = \{S, A, B\}$$
$$P = \begin{cases} S \to aA \mid \varepsilon \\ A \to aA \mid B \\ B \to bB \mid \varepsilon \end{cases}$$

   Veamos ahora que esta gramática no es ambigua. Sea $z \in L$.

   - Si $z = \varepsilon$, entonces la única derivación posible es $S \Rightarrow \varepsilon$.
   - Si $z \neq \varepsilon$, entonces $z = a^i b^j$, con $i, j \in \mathbb{N} \cup \{0\}$, $i \geqslant 1$. Para obtener la primera i (ya que $i \geqslant 1$) hemos de usar la producción $S \to aA$. A partir de aquí, hemos de usar la producción $A \to aA$ $i - 1$ veces. Por último, hemos de usar la producción $A \to B$ para producir las b's y la producción $B \to bB$ j veces. Por último, hemos de usar la producción $B \to \varepsilon$ para terminar. Por tanto, la única derivación posible es:

$$S \Rightarrow aA \Rightarrow aaA \Rightarrow \ldots \Rightarrow a^iA \Rightarrow a^iB \Rightarrow a^ibB \Rightarrow \ldots \Rightarrow a^ib^jB \Rightarrow a^ib^j$$

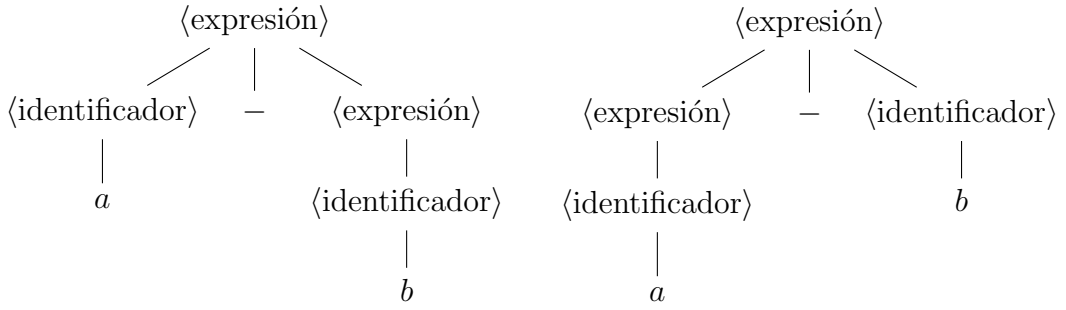


Figura 1.3: Árboles de derivación para $a - b$ usando la Gramática del Ejercicio 1.1.3.

Ejercicio 1.1.3. Considera la gramática $G = (V, T, S, P)$ donde

$$V = \{\langle\text{expresión}\rangle, \langle\text{identificador}\rangle\}$$
$$T = \{a, b, c, d, -\}$$
$$S = \langle\text{expresión}\rangle$$
$$P = \begin{cases} \langle\text{expresión}\rangle \to \langle\text{identificador}\rangle \\ \langle\text{expresión}\rangle \to \langle\text{identificador}\rangle - \langle\text{expresión}\rangle \\ \langle\text{expresión}\rangle \to \langle\text{expresión}\rangle - \langle\text{identificador}\rangle \\ \langle\text{identificador}\rangle \to a \mid b \mid c \mid d \end{cases}$$

1. Demuestra que esta gramática no puede ser empleada para describir un posible lenguaje de programación, teniendo en cuenta que la sustración no es una operación conmutativa, y que $(a - b) - d \neq a - (b - d)$.

   El lenguaje generado tiene como expresión regular:

   $$(a + b + c + d)\,(-(a + b + c + d))^*$$

   Por tanto, debido a que no tenemos paréntesis, en el lenguaje de programación no podríamos obtener el resultado de $a - (b - d)$.

2. ¿Es ambigua la gramática G? ¿Es la ambigüedad inherente al lenguaje generado por G? Justifica adecuadamente la respuesta.

   Para ver que la gramática es ambigua, consideramos la palabra $a - b$. Esta palabra tiene dos árboles de derivación, como se muestra en la Figura 1.3.

   La ambiguedad no es adherente al lenguaje generado por la gramática, ya que este lenguaje es regular. Por tanto, el lenguaje no es inherentemente ambiguo.

3. ¿Es posible modificar G de manera que la nueva gramática pueda ser usada para generar el lenguaje de las expresiones aritméticas correctas con el operador de resta?

   Sí, podemos añadir paréntesis a la gramática para que sea capaz de generar el lenguaje de las expresiones aritméticas correctas con el operador de resta. La gramática modificada sería $G = (V, \{a, b, c, d, -, (,)\}, P, S)$, con:

   $$V = \{S, I\}$$
   $$P = \begin{cases} I \to a \mid b \mid c \mid d \\ S \to I \mid (I - S) \mid (S - I) \end{cases}$$

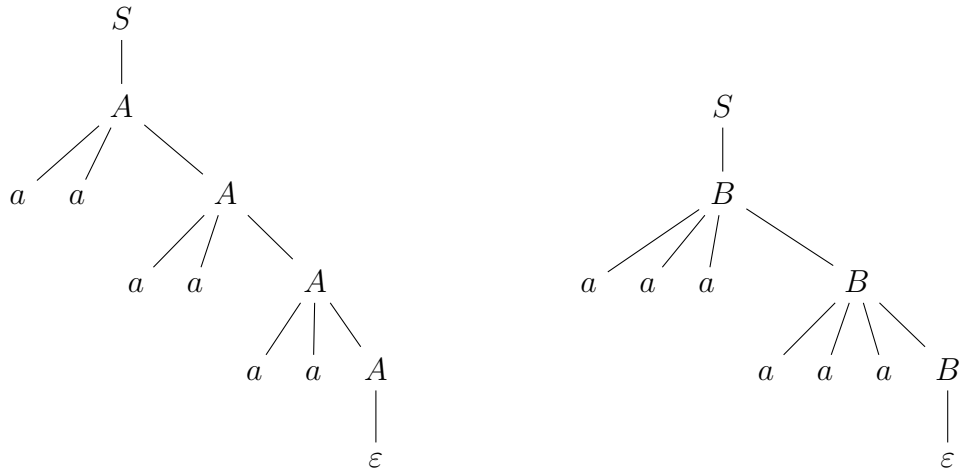


Figura 1.4: Árboles de derivación para a^6 usando la Gramática del Ejercicio 1.1.4.

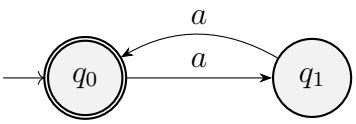


Figura 1.5: AFD que acepta el lenguaje de la variable A de la Gramática del Ejercicio 1.1.4.

Ejercicio 1.1.4. Dada la gramática

$$\begin{cases} S \to A \mid B \\ A \to aaA \mid \varepsilon \\ B \to aaaB \mid \varepsilon \end{cases}$$

1. Demostrar que es ambigua.

   La variable A genera palabras de la forma a^{2i} y la variable B genera palabras de la forma a^{3i}. Por tanto, la palabras de la forma a^{6i} tienen dos árboles de derivación, como se muestra en la Figura 1.4.

2. Construir un autómata finito determinístico que acepte el mismo lenguaje.

   El autómata que genera las palabras de la forma a^{2i} es el que se muestra en la Figura 1.5, mientras que el autómata que genera las palabras de la forma a^{3i} es el que se muestra en la Figura 1.6. El autómata producto es el que se muestra en la Figura 1.7.

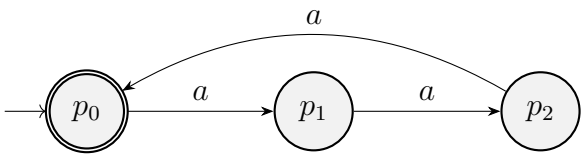


Figura 1.6: AFD que acepta el lenguaje de la variable B de la Gramática del Ejercicio 1.1.4.

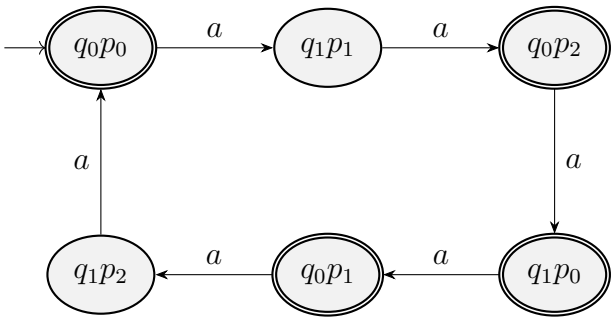


Figura 1.7: AFD que acepta el lenguaje de la Gramática del Ejercicio 1.1.4.

3. Construir una gramática lineal por la derecha, a partir del autómata determinístico, que genere el mismo lenguaje.

   La gramática lineal por la derecha que genera el lenguaje del autómata de la Figura 1.7 es $G = (V, \{a\}, P, S)$, con:

$$V = \{q_0p_0, q_1p_1, q_0p_2, q_1p_0, q_0p_1, q_1p_2\}$$

$$P = \begin{cases} q_0p_0 \to aq_1p_1 \mid \varepsilon \\ q_1p_1 \to aq_0p_2 \\ q_0p_2 \to aq_1p_0 \mid \varepsilon \\ q_1p_0 \to aq_0p_1 \mid \varepsilon \\ q_0p_1 \to aq_1p_2 \mid \varepsilon \\ q_1p_2 \to aq_0p_0 \end{cases}$$

4. Demostrar que la gramática resultante no es ambigua.

   Como el autómata del que proviene es determinista, la gramática obtenida no es ambigua; ya que para cada estado y símbolo solo hay un posible estado al que ir.

Ejercicio 1.1.5. Dar una gramática libre de contexto no ambigua que genere el lenguaje

$$L = \{a^ib^ja^kb^l \mid (i = j) \vee (k = l)\}$$

Este ejercicio no es tan directo y requiere explicación. La idea es expresar L como unión de tres lenguajes disjuntos:

1. $L_1 = \{a^ib^ja^kb^l \mid i, j \in \mathbb{N} \cup \{0\},\ k, l \in \mathbb{N} \setminus \{0\}, i = j \wedge k \neq l\}$
2. $L_2 = \{a^ib^ja^kb^l \mid i, j, k, l \in \mathbb{N} \setminus \{0\}, i \neq j \wedge k = l\}$
3. $L_3 = \{a^ib^ja^kb^l \mid i, j, k, l \in \mathbb{N} \setminus \{0\}, i = j \wedge k = l\} \cup \{\varepsilon\}$.

Vemos de forma directa que $L_1 = \bigcup_{i=1}^{3} L_i$. Veamos ahora que $\bigcap_{i=1}^{3} L_i = \emptyset$. Sea $z = a^ib^ja^kb^l \in L$. Si todos los exponentes son distintos de 0, entonces vemos de forma directa que tan solo puede pertenecer a uno de los L_i. Si $z = \varepsilon$, tan solo puede pertenecer a L_3. Por último, tan solo queda contemplar que la palabra sea de

la forma $a^p b^q$. En este caso, tan solo puede pertenecer a L_1.

Por tanto, tenemos que:

$$\bigcup_{i=1}^{3} L_i = L \qquad \bigcap_{i=1}^{3} L_i = \emptyset$$

Damos ahora una gramática para cada uno de los lenguajes.

1. Sea $G_1 = (V_1, \{a, b\}, P_1, S_1)$, con:

$$V_1 = \{S_1, A_1, B_1, B_1^a, B_2^b\}$$
$$P_1 = \begin{cases} S_1 \to A_1 B_1 \\ A_1 \to aA_1 b \mid \varepsilon \\ B_1 \to aB_1 b \mid B_1^a \mid B_2^b \\ B_1^a \to aB_1^a \mid a \\ B_2^b \to bB_2^b \mid b \end{cases}$$

   Notemos que A_1 genera la parte de la forma $a^i b^j$ con $i, j \in \mathbb{N} \cup \{0\}$, $i = j$. B_1 inicialmente genera la parte de la palabra de la forma $a^{k'} b^{l'}$ con $k', l' \in \mathbb{N} \cup \{0\}$, $k' = l'$, pero necesariamente se emplea B_1^a o B_2^b para terminar la palabra, y estas añaden respectivamente a's o b's (al menos una), de forma que se fuerza a que $k, l \in \mathbb{N} \setminus \{0\}$, $k \neq l$.

   Además, es no ambigua puesto que la la forma de la palabra fija qué producciones hemos de emplear.

   De esta forma, $\mathcal{L}(G_1) = L_1$.

2. Sea $G_2 = (V_2, \{a, b\}, P_2, S_2)$, con:

$$V_2 = \{S_2, A_2, B_2, A_2^a, A_2^b\}$$
$$P_2 = \begin{cases} S_2 \to A_2 B_2 \\ A_2 \to aA_2 b \mid A_2^a \mid A_2^b \\ B_2 \to aB_2 b \mid ab \\ A_2^a \to aA_2^a \mid a \\ A_2^b \to bA_2^b \mid b \end{cases}$$

   La demostración de que $\mathcal{L}(G_2) = L_2$ es análoga a la de G_1. Notemos que en este caso, como la regla $B_2 \to \varepsilon$ no está presente, se fuerza a que $k, l \neq 0$.

   Además, de igual forma, se trata de una gramática no ambigua.

   Por tanto, $\mathcal{L}(G_2) = L_2$.

3. Sea $G_3 = (V_3, \{a, b\}, P_3, S_3)$, con:

$$V_3 = \{S_3, C_3\}$$
$$P_3 = \begin{cases} S_3 \to C_3 C_3 \mid \varepsilon \\ C_3 \to aC_3 b \mid ab \end{cases}$$

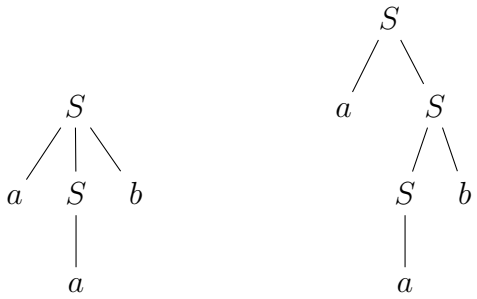


Figura 1.8: Árboles de derivación para *aab* usando la Gramática del Ejercicio 1.1.6.1.

Notemos que la regla $S \to \varepsilon$ se añade para que $\varepsilon \in L_3$. C_3 genera la parte de la palabra de la forma $a^i b^i$ con $i \in \mathbb{N} \setminus \{0\}$, por lo que se tiene.

Además, es no ambigua puesto que la la forma de la palabra fija qué producciones hemos de emplear.

Por tanto, $\mathcal{L}(G_3) = L_3$.

Como son tres lenguajes disjuntos cuya unión es L, y como cada una de las gramáticas es no ambigua, tenemos que la gramática $G = (V, \{a, b\}, P, S)$ es no ambigua, con:

$$V = V_1 \cup V_2 \cup V_3 \cup \{S\}$$
$$P = P_1 \cup P_2 \cup P_3 \cup \{S \to S_1 \mid S_2 \mid S_3 \quad \}$$

Ejercicio 1.1.6. Determinar cuales de las siguientes gramáticas son ambiguas y, en su caso, comprobar si los lenguajes generados son inherentemente ambiguos:

1. $S \to aSb \mid Sb \mid aS \mid a$

   La gramática dada es ambigua. Por ejemplo, la palabra *aab* tiene dos árboles de derivación, como se muestra en la Figura 1.8. No obstante, este es un lenguaje regular con expresión regular asociada:

   $$aa^*b^*$$

   Por tanto, el lenguaje generado no es inherentemente ambiguo.

2. $S \to aaS \mid aaaS \mid a$

   La gramática dada es ambigua. Por ejemplo, la palabra a^7 tiene dos árboles de derivación, como se muestra en la Figura 1.9. No obstante, este es un lenguaje regular con expresión regular asociada:

   $$(aa + aaa)^*a$$

   Por tanto, el lenguaje generado no es inherentemente ambiguo.

3. $S \to aS \mid aSb \mid X$,
   $X \to Xa \mid a$

   La gramática dada es ambigua. Por ejemplo, la palabra a^2 tiene dos árboles de derivación, como se muestra en la Figura 1.10.

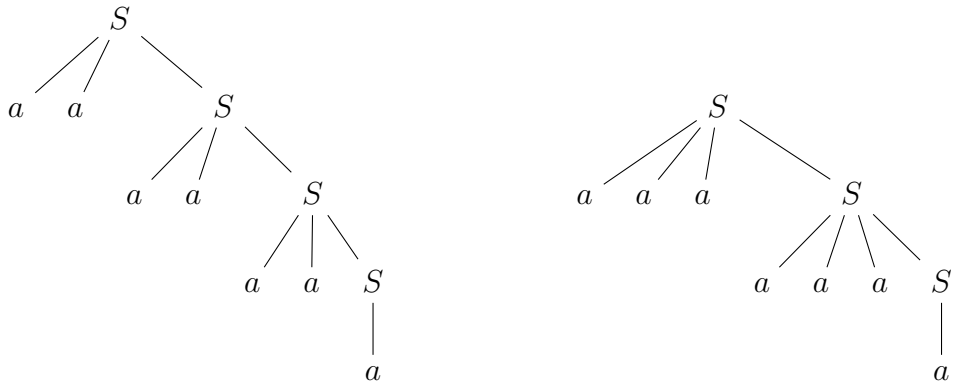


Figura 1.9: Árboles de derivación para a^7 usando la Gramática del Ejercicio 1.1.6.2.

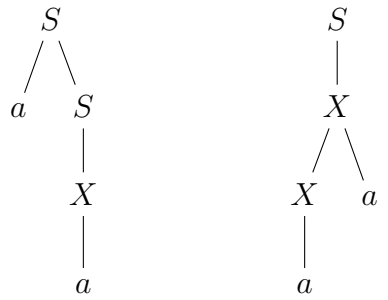


Figura 1.10: Árboles de derivación para a^2 usando la Gramática del Ejercicio 1.1.6.3.

El lenguaje generado por esta gramática es:

$$L = \{aa^{n+m}b^n \mid n, m \in \mathbb{N} \cup \{0\}\}$$

Consideramos ahora la gramática $G = (V, \{a, b\}, P, S)$, con:

$$V = \{S, A, B\}$$
$$P = \begin{cases} S \to aA \\ A \to aA \mid B \\ B \to aBb \mid \varepsilon \end{cases}$$

Esta gramática no es ambigua. Sea $z \in L$, por lo que será de la forma $z = aa^{n+m}b^n$. Para obtener la primera a hemos de usar la producción $S \to aA$. A partir de aquí, hemos de usar la producción $A \to aA$ m veces. Por último, hemos de usar la producción $A \to B$ para producir las a's y b's y la producción $B \to aBb$ n veces. Por último, hemos de usar la producción $B \to \varepsilon$ para terminar. Por tanto, la única derivación posible es:

$$S \Rightarrow aA \Rightarrow aaA \Rightarrow \ldots \Rightarrow aa^mB \Rightarrow aa^{n+m}b^nB \Rightarrow aa^{n+m}b^n$$

Ejercicio 1.1.7. Dar gramáticas libres de contexto no ambiguas (cuando sea posible) pa2ra los siguientes lenguajes sobre el alfabeto $A = \{a, b, c\}$:

1. $L_1 = \{a^ib^jc^k \mid i \neq j \vee j \neq k\}$

2. $L_2 = \{(ab)^i(bc)^j \mid i, j \geqslant 0\}$

   La gramática que genera el lenguaje L_2 es $G = (V, \{a, b, c\}, P, S)$, con:

$$V = \{S, X\}$$
$$P = \begin{cases} S \to abS \mid X \\ X \to bcX \mid \varepsilon \end{cases}$$

   Esta no es ambigua. Sea $z \in L$, por lo que será de la forma $z = (ab)^i(bc)^j$. Para obtener la parte de $(ab)^i$ hemos de usar la producción $S \to abS$ i veces. A partir de aquí, hemos de usar la producción $S \to X$ para producir las b's y c's y la producción $X \to bcX$ j veces. Por último, hemos de usar la producción $X \to \varepsilon$ para terminar. Por tanto, la única derivación posible es:

$$S \Rightarrow abS \Rightarrow ababS \Rightarrow \ldots \Rightarrow (ab)^iX \Rightarrow (ab)^i(bc)X \Rightarrow (ab)^i(bc)^jX \Rightarrow (ab)^i(bc)^j$$

3. $L_3 = \{a^ib^{i+j}c^j \mid i, j \geqslant 0\}$

   La gramática que genera el lenguaje L_3 es $G = (V, \{a, b, c\}, P, S)$, con:

$$V = \{S, A, C\}$$
$$P = \begin{cases} S \to AC \\ A \to aAb \mid \varepsilon \\ C \to bCc \mid \varepsilon \end{cases}$$

   Esta no es ambigua. Sea $z \in L$, por lo que será de la forma $z = a^ib^{i+j}c^j$. Para obtener la parte de a^ib^i hemos de usar la producción $A \to aAb$ i veces. A partir de aquí, hemos de usar la producción $A \to \varepsilon$. Por otro lado, para obtener la parte de b^jc^j hemos de usar la producción $C \to bCc$ j veces. Por último, hemos de usar la producción $C \to \varepsilon$ para terminar.

4. L_4 definido como el conjunto de palabras que comienzan por aab y terminan por bbc y tales que estas dos subcadenas no aparecen nunca en el interior de la palabra (sólo están al principio y al final).

   Notemos que, como el enunciado no es totalmente preciso, consideramos $aabbc \notin L_4$. El AFD que acepta el lenguaje L_4 es el de la Figura 1.11.

   La gramática que genera el lenguaje L_4 es $G = (V, \{a, b, c\}, P, q_0)$, con:

$$V = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$$
$$P = \begin{cases} q_0 \to aq_1 \\ q_1 \to aq_2 \\ q_2 \to bq_3 \\ q_3 \to aq_7 \mid bq_4 \mid cq_3 \\ q_4 \to aq_7 \mid bq_5 \mid cq_3 \\ q_5 \to bq_5 \mid cq_6 \mid aq_7 \\ q_6 \to \varepsilon \\ q_7 \to aq_8 \mid bq_4 \mid cq_3 \\ q_8 \to aq_8 \mid cq_3 \end{cases}$$

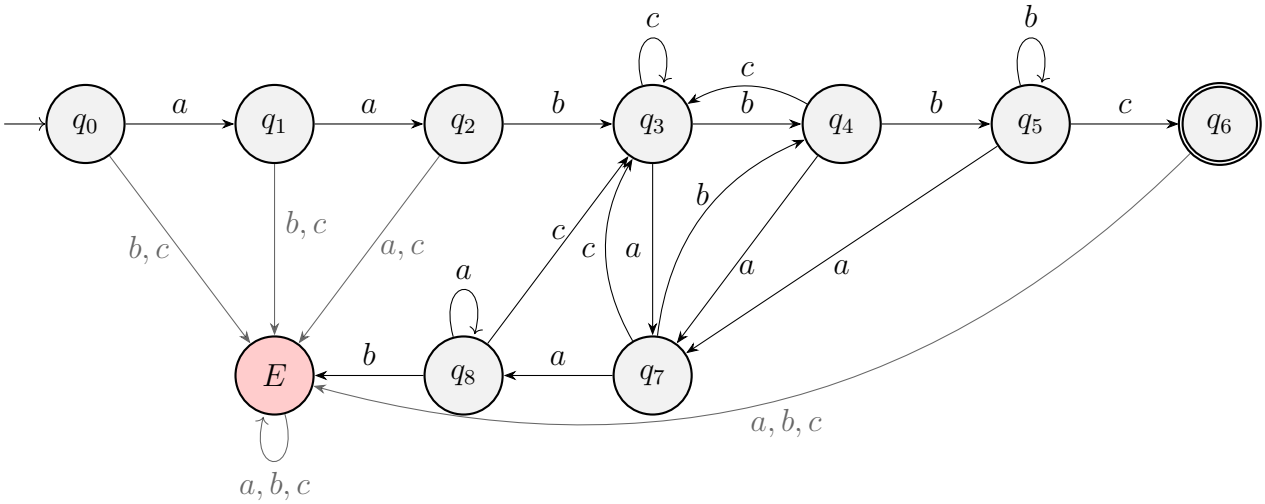


Figura 1.11: AFD que acepta el lenguaje L_4.

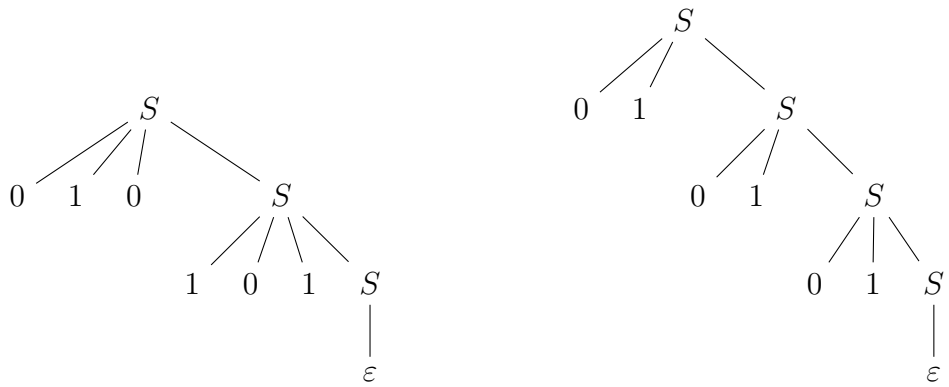


Figura 1.12: Árboles de derivación para 010101 usando la Gramática del Ejercicio 1.1.8.

donde no hemos introducido la variable E ni las producciones asociadas con ella debido a que esta variable es inútil. Esta gramática es no ambigua puesto que proviene de un AFD.

Ejercicio 1.1.8. Dada la gramática

$$\{S \to 01S \mid 010S \mid 101S \mid \varepsilon$$

1. Determinar si es ambigua.

   La gramática dada es ambigua. Por ejemplo, la palabra 010101 tiene dos árboles de derivación, como se muestra en la Figura 1.12.

2. Construir un autómata finito determinista asociado.

   La expresión regular asociada al lenguaje generado por la gramática es:

   $$(01 + 010 + 101)^*$$

   El AFND asociado es el de la Figura 1.13.

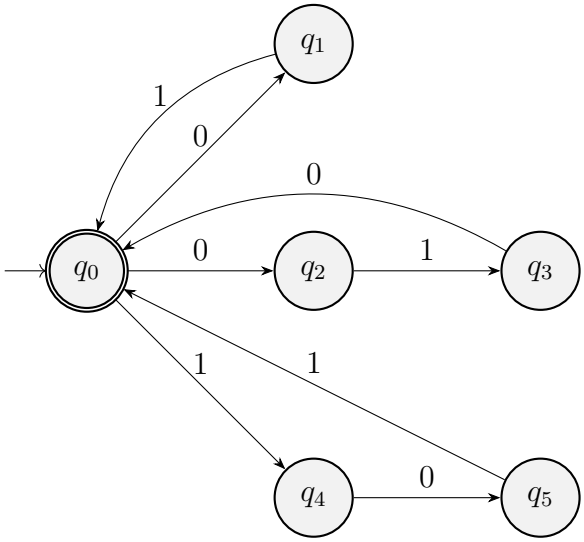


Figura 1.13: AFND que acepta el lenguaje de la Gramática del Ejercicio 1.1.8.

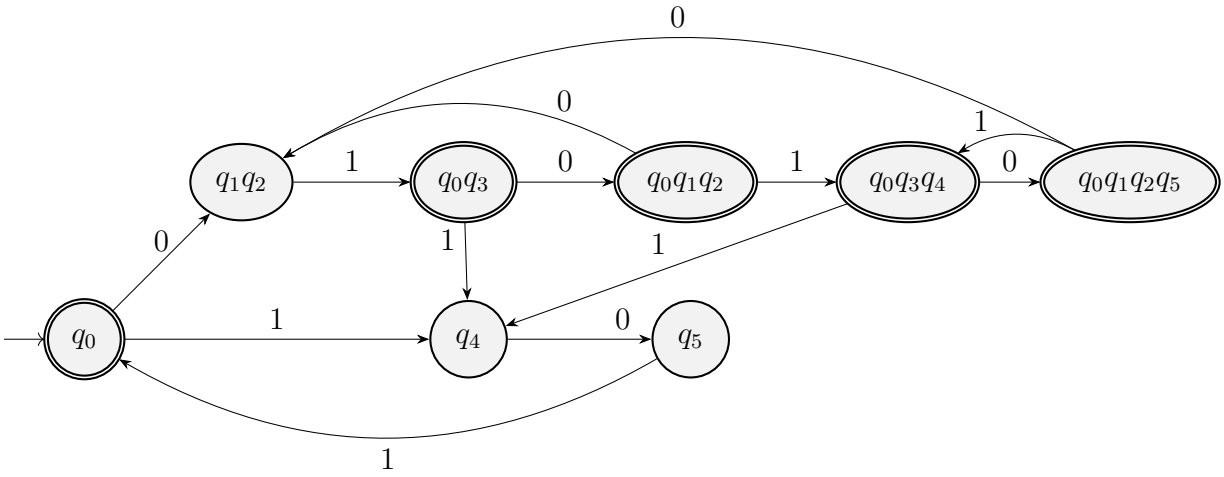


Figura 1.14: AFD que acepta el lenguaje de la Gramática del Ejercicio 1.1.8.

El AFD asociado es el de la Figura 1.14, donde las transiciones que faltan son transiciones a un estado de error. Este no se añade para evitar confusión, y no afectará al siguiente apartado.

3. Calcular la gramática lineal por la derecha que se obiene a partir del autómata. ¿Es ambigua la gramática resultante?

La gramática lineal por la derecha que se obtiene a partir del AFD es $G =$

$(V, \{0,1\}, P, q_0)$, con:

$$V = \{q_0, q_1q_2, q_0q_3, q_0q_1q_2, q_0q_3q_4, q_0q_1q_2q_5, q_4, q_5\}$$

$$P = \begin{cases} q_0 \to 0q_1q_2 \mid 1q_4 \mid \varepsilon \\ q_1q_2 \to 1q_0q_3 \\ q_0q_3 \to 1q_4 \mid 0q_0q_1q_2 \mid \varepsilon \\ q_0q_1q_2 \to 0q_1q_2 \mid 1q_0q_3q_4 \mid \varepsilon \\ q_0q_3q_4 \to 0q_0q_1q_2q_5 \mid 1q_4 \mid \varepsilon \\ q_0q_1q_2q_5 \to 0q_1q_2 \mid 1q_0q_3q_4 \mid \varepsilon \\ q_4 \to 0q_5 \\ q_5 \to 1q_0 \end{cases}$$

Esta es no ambigua, puesto que proviene de un AFD.

Ejercicio 1.1.9. Considerar la siguiente gramática:

$$\begin{cases} S \to A1B \\ A \to 0A \mid \varepsilon \\ B \to 0B \mid 1B \mid \varepsilon \end{cases}$$

1. Demostrar que la gramática dada no es ambigua.

   La expresión regular asociada al lenguaje generado por la gramática es:

   $$0^*1(0+1)^*$$

   La gramática dada no es ambigua. Sea $z \in L$, por lo que será de la forma $z = 0^i1u$, con $i \geqslant 0$ y $u \in \{0,1\}^*$. En primer lugar, para obtener el 1 central, hemos de usar la producción $S \to A1B$. A partir de aquí, hemos de usar la producción $A \to 0A$ i veces. Después, usamos las reglas de B para obtener la palabra u.

2. Encontrar una gramática para el mismo lenguaje que sea ambigua y demostrar su ambigüedad.

   Consideramos la gramática $G = (V, \{0,1\}, P, S)$, con:

   $$V = \{S, A\}$$
   $$P = \begin{cases} S \to 0S \mid 1A \mid 1 \\ A \to 0A \mid 1A \mid \varepsilon \end{cases}$$

   Notemos que esta es ambigua, puesto que si consideramos la palabra 1, esta tiene dos árboles de derivación, como se muestra en la Figura 1.15. Notemos que si quitamos la regla $S \to 1$, la gramática no sería ambigua y generaría el mismo lenguaje que la gramática original.

Ejercicio 1.1.10. Considerar la siguiente gramática $G = (\{S, A\}, \{a, b\}, P, S)$, con

$$P = \begin{cases} S \to aAa \mid bAa \\ A \to aAa \mid bAa \mid \varepsilon \end{cases}$$

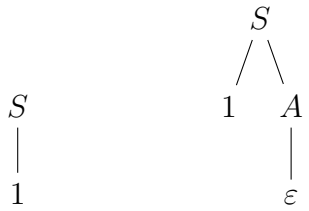


Figura 1.15: Árboles de derivación para 1 usando la Gramática del Ejercicio 1.1.9.

1. Describir el lenguaje generado por la gramática.

   Tenemos que:

   $$\mathcal{L}(G) = \{ua^n \mid n \in \mathbb{N} \setminus \{0\}, u \in \{a,b\}^*,\ |u| = n\}$$

2. Demuestra que el lenguaje generado por la gramática no es regular, pero sí independiente del contexto.

   Tenemos que es independiente del conexto, pero no sabemos si es regular o no. Para demostrar que no es regular, usamos el Lema de Bombeo. Para cada $n \in \mathbb{N}$, consideramos la palabra $z = b^n a^n$, con $|z| = 2n \geqslant n$. Entonces, para toda descompisición $z = uvw$ con $u, v, w \in \{a,b\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ se tiene que:

   $$u = b^k, \quad v = b^l, \quad w = b^{n-k-l}a^n \quad \text{con } 0 \leqslant k + l \leqslant n,\ l \geqslant 1$$

   Entonces, para $i = 2$, tenemos que $uv^2w = b^{k+2l+n-k-l}a^n = b^{n+l}a^n \notin L$, ya que $n + l \neq n$. Por tanto, por el recíproco del Lema de Bombeo, el lenguaje no es regular.

3. Normaliza la gramática G en la Forma Normal de Greibach, y determina todas las derivaciones más a la izquierda para la cadena ab^2a^5.

   En primer lugar, vemos que todas las producciones son útiles. Buscamos ahora eliminar las producciones nulas. Para esto, vemos que la única producción variable anulable es A. Eliminando las producciones nulas, obtenemos las siguientes reglas de producción:

   $$P = \begin{cases} S \to aAa \mid bAa \mid aa \mid ba \\ A \to aAa \mid bAa \mid aa \mid ba \end{cases}$$

   Para que esté en forma normal de Greibach, necesitamos que todas las reglas sean de la forma $A \to a\alpha$, con $a \in T$, $\alpha \in V^*$. Por tanto, las producciones resultantes para que esté en forma normal de Greibach son:

   $$P = \begin{cases} S \to aAC_a \mid bAC_a \mid aC_a \mid bC_a \\ A \to aAC_a \mid bAC_a \mid aC_a \mid bC_a \\ C_a \to a \end{cases}$$

   Esta gramática, efectivamente, esta en forma normal de Greibach. Para la cadena ab^2a^5, las derivaciones más a la izquierda son:

   $S \Rightarrow aAC_a \Rightarrow abAC_aC_a \Rightarrow abbAC_aC_aC_a \Rightarrow abbaC_aC_aC_aC_a \Rightarrow abbaC_aC_aC_aC_a \Rightarrow ab^2a^5$

Ejercicio 1.1.11. Obtener la forma normal de Greibach para la siguiente gramática:

$$G = \{\{S_1, S_2, S_3\}, \{a, b, c, d, e\}, S_1, P\}$$

donde:

$$P = \begin{cases} S_1 \to S_1 S_2 c \mid S_3 \mid S_3 b S_3 \\ S_2 \to S_1 S_1 \mid d \\ S_3 \to S_2 e \end{cases}$$

Vemos que esta gramática no tiene producciones nulas. Eliminamos las producciones unitarias. Para esto, vemos que las únicas producciones unitarias son $S_1 \to S_3$ y $S_3 \to S_2 e$. Por tanto, eliminamos estas producciones unitarias y obtenemos las siguientes reglas de producción:

$$P = \begin{cases} S_1 \to S_1 S_2 c \mid S_2 e \mid S_3 b S_3 \\ S_2 \to S_1 S_1 \mid d \\ S_3 \to S_2 e \end{cases}$$

Eliminamos ahora los símbolos terminales de las producciones que no son de la forma $A \to z$, $z \in T$. Para esto, introducimos variables auxiliares. Las producciones resultantes son:

$$P = \begin{cases} S_1 \to S_1 S_2 C_c \mid S_2 C_e \mid S_3 C_b S_3 \\ S_2 \to d \mid S_1 S_1 \\ S_3 \to S_2 C_e \\ C_b \to b \\ C_c \to c \\ C_e \to e \end{cases}$$

Eliminamos ahora las producciones de la forma $A \to B_1 B_2 \dots B_n$, con $n \geqslant 3$. Para esto, introducimos variables auxiliares. Las producciones resultantes son:

$$P = \begin{cases} S_1 \to S_1 D_1 \mid S_2 C_e \mid S_3 D_2 \\ S_2 \to d \mid S_1 S_1 \\ S_3 \to S_2 C_e \\ C_b \to b \\ C_c \to c \\ C_e \to e \\ D_1 \to S_2 C_c \\ D_2 \to C_b S_3 \end{cases}$$

Ejercicio 1.1.12. Pasar a forma normal de Greibach la gramática

$$\begin{cases} S \to AAA \mid B \\ A \to aA \mid B \\ B \to \varepsilon \end{cases}$$

Obtenemos en primer lugar las variables anulables, que son $H = \{S, B, A\}$. Como $S \in H$, tenemos que $\varepsilon \in \mathcal{L}(G)$. Tras aliminar las producciones nulas y añadir las producciones correspondientes, obtenemos las siguientes reglas de producción:

$$\begin{cases} S \to AAA \mid AA \mid A \mid B \\ A \to aA \mid a \mid B \end{cases}$$

Como B es una variable inútil, eliminamos las producciones que contienen a B y obtenemos las siguientes reglas de producción:

$$\begin{cases} S \to AAA \mid AA \mid A \\ A \to aA \mid a \end{cases}$$

Eliminamos ahora las producciones unitarias ($S \to A$), y obtenemos las siguientes reglas de producción:

$$\begin{cases} S \to AAA \mid AA \mid aA \mid a \\ A \to aA \mid a \end{cases}$$

Ejercicio 1.1.13. Determina si los siguientes lenguajes son regulares o independientes del contexto. Encuentra una gramática que los genere.

1. $L_1 = \{a^i b^j c^k \mid i, j \geqslant 0, k < i + j\}$

   Veamos que no es regular usando el Lema de Bombeo. Para cada $n \in \mathbb{N}$, consideramos la palabra $z = a^n b^n c^{2n-1}$, con $|z| = 4n - 1 \geqslant n$. Entonces, para toda descomposición $z = uvw$ con $u, v, w \in \{a, b, c\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ se tiene que:

   $$u = a^k, \quad v = a^l, \quad w = a^{n-k-l} b^n c^{2n-1} \quad \text{con } 0 \leqslant k + l \leqslant n,\ l \geqslant 1$$

   Entonces, para $i = 0$, tenemos que $uv^0w = a^{k+n-k-l} b^n c^n = a^{n-l} b^n c^{2n-1} \notin L$, ya que:

   $$2n - 1 < 2n - l \iff -1 < -l \iff l < 1$$

   Por tanto, llegamos a una contradicción, por lo que L_1 no es regular. Veamos que es independiente del contexto. Para esto, vemos en el lenguaje descrito que, por cada c, hay un a o un b. Además, como la desigualdad es estricta, hay al menos un a o un b (o ambos) que no está balanceado con un c. Consideramos por tanto los siguientes lenguajes, que nos serán de ayuda:

   a) L_{aux}. Se da la igualdad. Cada c está balanceada con un a o un b, y cada a y cada b están balanceados con un c.

   b) L_{1a}. $k < i + j$. Cada c está balanceada con un a o un b. Cada b está balanceada con un c, y al menos un a no está balanceado con un c.

   c) L_{1b}. $k < i + j$. Cada c está balanceada con un a o un b. Cada a está balanceada con un c, y al menos un b no está balanceado con un c.

   d) L_{1ab}. $k < i + j$. Cada c está balanceada con un a o un b. Al menos un a no está balanceado con un c, y al menos un b no está balanceado con un c.

Vemos que $L_{1a} \cap L_{1b} \cap L_{1ab} = \emptyset$, y que $L_1 = L_{1a} \cup L_{1b} \cup L_{1ab}$. Veamos en primer lugar las gramáticas que generan cada uno de estos lenguajes:

Lenguaje auxiliar L_{aux}) Este lenguaje es:

$$L_{aux} = \{a^i b^j c^k \mid i, j, k \geqslant 0, k = i + j\}$$

Notemos que este es una primera aproximación, donde cada c está balanceada con un a o un b y, además, cada a y cada b están balanceados con un c. La gramática que genera este lenguaje es:

$$\begin{aligned} G_{aux} &= (V_{aux}, \{a, b, c\}, P_{aux}, S_{aux}) \\ V_{aux} &= \{S_{aux}, X_{aux}\} \\ P &= \begin{cases} S_{aux} \to aS_{aux}c \mid X_{aux} \\ X_{aux} \to bX_{aux}c \mid \varepsilon \end{cases} \end{aligned}$$

Esta gramática no nos es estrictamente necesaria, pero nos será de ayuda para obtener las gramáticas que generan los lenguajes L_{1a}, L_{1b} y L_{1ab}. Notemos que $S \to aSc$ genera un a balanceada con un c, y $X \to bXc$ genera un b balanceado con un c. Estas dos reglas habrán de mantenerse para que cada c esté balanceada con un a o un b.

Lenguaje L_{1a}) En este lenguaje, se mantiene que cada c está balanceada con un a o un b y que cada b está balanceada con un c. Sin embargo, al menos un a no está balanceado con un c. La gramática que genera este lenguaje es:

$$\begin{aligned} G_{1a} &= (V_{1a}, \{a, b, c\}, P_{1a}, S_{1a}) \\ V_{1a} &= \{S_{1a}, X_{1a}, A_{1a}\} \\ P_{1a} &= \begin{cases} S_{1a} \to aS_{1a}c \mid A_{1a}X_{1a} \\ X_{1a} \to bX_{1a}c \mid \varepsilon \\ A_{1a} \to aA_{1a} \mid a \end{cases} \end{aligned}$$

Lenguaje L_{1b}) En este lenguaje, se mantiene que cada c está balanceada con un a o un b y que cada a está balanceada con un c. Sin embargo, al menos un b no está balanceado con un c. La gramática que genera este lenguaje es:

$$\begin{aligned} G_{1b} &= (V_{1b}, \{a, b, c\}, P_{1b}, S_{1b}) \\ V_{1b} &= \{S_{1b}, X_{1b}, B_{1b}\} \\ P_{1b} &= \begin{cases} S_{1b} \to aS_{1b}c \mid X_{1b} \\ X_{1b} \to bX_{1b}c \mid B_{1b} \\ B_{1b} \to bB_{1b} \mid b \end{cases} \end{aligned}$$

Lenguaje L_{1ab}) En este lenguaje, se mantiene que cada c está balanceada con un a o un b. Sin embargo, al menos un a no está balanceado con un c y al menos un b no está balanceado con un c. La gramática que genera

este lenguaje es:

$$
\begin{aligned}
G_{1ab} &= (V_{1ab}, \{a,b,c\}, P_{1ab}, S_{1ab}) \\
V_{1ab} &= \{S_{1ab}, X_{1ab}, A_{1ab}, B_{1ab}\} \\
P_{1ab} &= \begin{cases} S_{1ab} \to aS_{1ab}c \mid A_{1ab}X_{1ab} \\ X_{1ab} \to bX_{1ab}c \mid B_{1ab} \\ A_{1ab} \to aA_{1ab} \mid a \\ B_{1ab} \to bB_{1ab} \mid b \end{cases}
\end{aligned}
$$

Como $L_1 = L_{1a} \cup L_{1b} \cup L_{1ab}$ y son disjuntos, la gramática que genera L_1 es la "unión" de las gramáticas que generan L_{1a}, L_{1b} y L_{1ab}, que describimos a continuación:

$$
\begin{aligned}
G &= (V, \{a,b,c\}, P, S) \\
V &= V_{1a} \cup V_{1b} \cup V_{1ab} \cup \{S\} \\
P &= P_{1a} \cup P_{1b} \cup P_{1ab} \cup \left\{ S \to S_{1a} \mid S_{1b} \mid S_{1ab} \quad \right\}
\end{aligned}
$$

A modo de resumen, y reduciendo el número de variables, la gramática que genera L_1 es:

$$
\begin{aligned}
G &= (V, \{a,b,c\}, P, S) \\
V &= \{S, A, B, X_{1a}, X_{1b}, X_{1ab}\} \\
P &= \begin{cases} S \to aSc \mid AX_{1a} \mid AX_{1ab} \mid X_{1b} \\ A \to aA \mid a \\ B \to bB \mid b \\ X_{1a} \to bX_{1a}c \mid \varepsilon \\ X_{1b} \to bX_{1b}c \mid B \\ X_{1ab} \to bX_{1ab}c \mid B \end{cases}
\end{aligned}
$$

Notemos además que esta gramática es no ambigua, por lo que L_1 no es inherentemente ambiguo. Además, como $\mathcal{L}(G) = L_1$, L_1 es independiente del contexto.

2. $L_2 = \{(ab)^i c^j d \mid j = i-1, i \geqslant 1\}$

Opción 1) Usando de forma directa el Lema de Bombeo.

Veamos que no es regular usando el Lema de Bombeo. Para cada $n \in \mathbb{N}$, consideramos la palabra $z = (ab)^n c^{n-1} d$, con $|z| = 2n+n-1+1 = 3n \geqslant n$. Entonces, para toda descomposición $z = uvw$ con $u, v, w \in \{a,b,c\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$, hay varias posibilidades:

- $u = (ab)^k$, $v = (ab)^l$ y $w = (ab)^{n-k-l} c^{n-1} d$ con $0 \leqslant 2k+2l \leqslant n$, $l \geqslant 1$.
- $u = (ab)^k$, $v = (ab)^l a$ y $w = b(ab)^{n-k-l-1} c^{n-1} d$ con $0 \leqslant 1+2k+2l \leqslant n$, $l \geqslant 0$.
- $u = (ab)^k a$, $v = b(ab)^l$ y $w = (ab)^{n-k-l-1} c^{n-1} d$ con $0 \leqslant 2+2k+2l \leqslant n$, $l \geqslant 0$.

- $u = (ab)^k a$, $v = b(ab)^l a$ y $w = b(ab)^{n-k-l-2}c^{n-1}d$ con $0 \leqslant 3+2k+2l \leqslant n$, $l \geqslant 0$.

Como vemos, hay muchas casuísticas a considerar, por lo que consideramos la siguiente opción.

Opción 2) Empleando que, si L_2 es regular, entonces L_2^{-1} es regular.

Supongamos por reducción al absurdo que L_2 es regular. Entonces, L_2^{-1} es regular, que es:

$$L_2^{-1} = \{dc^j(ab)^i \mid j = i-1, i \geqslant 1\}$$

Veamos que L_2^{-1} no es regular usando el Lema de Bombeo. Para cada $n \in \mathbb{N}$, consideramos la palabra $z = dc^{n-1}(ab)^n$, con $|z| = 1 + n - 1 + 2n = 3n \geqslant n$. Entonces, para toda descomposición $z = uvw$ con $u, v, w \in \{a, b, c\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$, hay varias posibilidades:

a) $u = dc^k$, $v = c^l$ y $w = c^{n-1-k-l}(ab)^n$ con $0 \leqslant 1 + k + l \leqslant n$, $l \geqslant 1$.
En este caso, si $i = 2$, tenemos que:

$$uv^2w = dc^k c^{2l} c^{n-1-k-l}(ab)^n = dc^{k+2l+n-1-k-l}(ab)^n = dc^{n+l-1}(ab)^n \notin L_2^{-1}$$

ya que $n + l - 1 \neq n - 1$ puesto que $l \geqslant 1$.

b) $u = \varepsilon$, $v = dc^{n-1}$ y $w = (ab)^n$.
En este caso, si $i = 0$, tenemos que:

$$uv^0w = (ab)^n \notin L_2^{-1}$$

puesto que no comienza por d.

Por tanto, por el recíproco del Lema de Bombeo, L_2^{-1} no es regular, llegando a una contradicción y por tanto L_2 no es regular.

La gramática que genera L_2 es:

$$G = (V, \{a, b, c, d\}, P, S)$$
$$V = \{S, A, B, C\}$$
$$P = \begin{cases} S \to Xd \\ X \to abXc \mid ab \end{cases}$$

Notemos que la producción $X \to ab$ provoca ese par ab que no está balanceado con un c. Por tanto, esta gramática genera L_2, y como se trata de una gramática independiente del contexto, L_2 es independiente del contexto.

3. $L_3 = \{ab^i cd^j \mid j = 2 \cdot i, 1 \leqslant i \leqslant 10\}$

Dado $z \in L_3$, tenemos que:

$$|z| \leqslant |ab^{10}cd^{2\cdot 20}| = 1 + 10 + 1 + 2 \cdot 20 = 52$$

Por tanto, se trata de un lenguaje finito, y por tanto regular. Sea este lenguaje:

$$L_3 = \{ab^1cd^2, ab^2cd^4, ab^3cd^6, ab^4cd^8, ab^5cd^{10}, ab^6cd^{12}, ab^7cd^{14}, ab^8cd^{16}, ab^9cd^{18}, ab^{10}cd^{20}\}$$

Por tanto, la gramática que genera L_3 es:

$$\begin{aligned} G &= (V, \{a, b, c, d\}, P, S) \\ V &= \{S\} \\ P &= \left\{ S \to ab^i cd^{2i} \mid i = 1, 2, \ldots, 10 \right. \end{aligned}$$

Elige una de ellas que sea independiente del contexto y pásala a forma normal de Chomsky.

Consideramos la gramática que genera L_2:

$$\begin{aligned} G &= (V, \{a, b, c, d\}, P, S) \\ V &= \{S, A, B, C\} \\ P &= \begin{cases} S \to Xd \\ X \to abXc \mid ab \end{cases} \end{aligned}$$

Eliminamos en lugar los símbolos terminales de las producciones que no son de la forma $A \to z$, $z \in T$. Para esto, introducimos variables auxiliares. La gramática resultante es:

$$\begin{aligned} G &= (V, \{a, b, c, d\}, P, S) \\ V &= \{S, A, B, C, X_1, X_2\} \\ P &= \begin{cases} S \to XC_d \\ X \to C_aC_bXC_c \mid C_aC_b \\ C_a \to a \\ C_b \to b \\ C_c \to c \\ C_d \to d \end{cases} \end{aligned}$$

Para eliminar las producciones de la forma $A \to B_1B_2 \ldots B_n$, con $n \geqslant 3$, introducimos variables auxiliares. La gramática resultante es:

$$\begin{aligned} G &= (V, \{a, b, c, d\}, P, S) \\ V &= \{S, A, B, C, X_1, X_2\} \\ P &= \begin{cases} S \to XC_d \\ X \to C_aD_1 \mid C_aC_b \\ D_1 \to C_bD_2 \\ D_2 \to XC_c \\ C_a \to a \\ C_b \to b \\ C_c \to c \\ C_d \to d \end{cases} \end{aligned}$$

Ejercicio 1.1.14. Dadas las siguientes gramáticas determinar si son ambiguas y, en caso de que lo sean, determinar una gramática no ambigua que genere el mismo lenguaje.

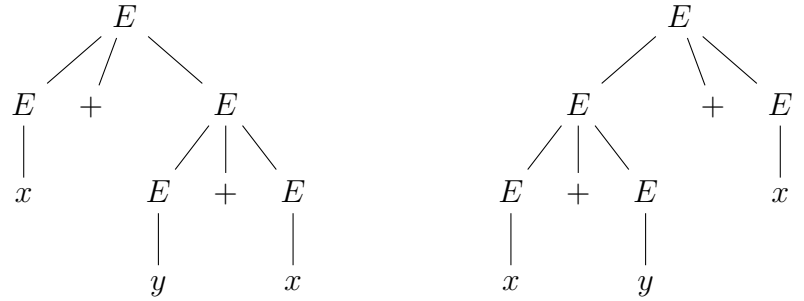


Figura 1.16: Árboles de derivación para $x + y + x$ usando la Gramática del Ejercicio 1.1.14.

1. $E \to E{+}E \mid E{*}E \mid (E) \mid x \mid y$ (alfabeto de símbolos terminales $\{x, y, +, *, (,)\}$ y símbolo inicial E).

   Esta gramática es ambigua. Por ejemplo, la palabra $x+y+x$ tiene dos árboles de derivación, como se muestra en la Figura 1.16.

2. $S \to SS{+} \mid SS{*} \mid x \mid y$ (alfabeto de símbolos terminales $\{x, y, +, *\}$ y símbolo inicial S)

Ejercicio 1.1.15. Una gramática independiente del contexto generalizada es una gramática en el que las producciones son de la forma $A \to r$ donde r es una expresión regular de variables y símbolos terminales. Una gramática independiente del contexto generalizada representa una forma compacta de representar una gramática con todas las producciones $A \to \alpha$, donde α es una palabra del lenguaje asociado a la expresión regular r y $A \to r$ es una producción de la gramática generalizada. Observemos que esta gramática asociada puede tener infinitas producciones, ya que una expresión regular puede representar un lenguaje con infinitas palabras. El concepto de lenguaje generado por una gramática generalizada se define de forma análoga al de las gramáticas independientes del contexto, pero teniendo en cuenta que ahora puede haber infinitas producciones. Demostrar que un lenguaje es independiente del contexto si y solo si se puede generar por una gramática generalizada.

Demostraremos por doble implicación.

$\Longrightarrow$) Supongamos que L es independiente del contexto. Entonces, existe una gramática independiente del contexto $G = (V, T, P, S)$ que genera L. Como es independiente del contexto, cada regla de producción será de la forma:

$$A \to \alpha, \qquad \text{con } A \in V,\ \alpha \in (V \cup T)^*$$

Como α es una sucesión de variables y símbolos terminales, podemos considerar que α es una expresión regular. Por tanto, G es una gramática generalizada.

$\Longleftarrow$) Supongamos que L es generado por una gramática generalizada $G = (V, T, P, S)$. Entonces, cada regla de producción será de la forma:

$$A \to r, \qquad \text{con } A \in V,\ r \text{ expresión regular}$$

Para cada una de las reglas de producción, distinguimos casos:

- Si $r = \emptyset$, eliminamos esa regla de producción.
- Si $r = \varepsilon$, esta es una regla aceptada en una gramática independiente del contexto.
- Si $r = a$, con $a \in T$, esta es una regla aceptada en una gramática independiente del contexto.
- Si $r = s + t$, con s, t expresiones regulares, entonces la eliminamos y añadimos la regla $A \to s \mid t$.
- Si $r = st$, con s, t expresiones regulares, entonces la eliminamos y añadimos las variables auxiliares $\{A_1, A_2\}$ y las reglas $A \to A_1A_2$, $A_1 \to s$, $A_2 \to t$.
- Si $r = s^*$, con s expresión regular, entonces la eliminamos y añadimos la variable auxiliar $\{B\}$ y las reglas $A \to B$, $B \to sB \mid \varepsilon$, donde la B ha sido necesaria para asegurarse de que se cumple la clausura de Kleene.

Tras aplicar el algoritmo por primera vez, llegamos a otra gramática generalizada, a la que le volvemos a aplicar el algoritmo. Repitiendo este proceso, llegamos a una gramática independiente del contexto que genera L, por lo que L es independiente del contexto.

Ejercicio 1.1.16. Demostrar que los siguientes lenguajes son independientes del contexto:

1. $L_1 = \{u\#w \mid u^{-1}$ es una subcadena de $w,\ u, w \in \{0,1\}^*\}$.

   Sea $z \in L_1$. Entonces, $z = u\#v_1u^{-1}v_2$, con $u, v_1, v_2 \in \{0,1\}^*$. Consideramos ahora la gramática $G = (V, \{0,1,\#\}, P, S)$, con:

   $$V = \{S, A, B\}$$
   $$P = \begin{cases} S \to S0 \mid S1 \mid A \\ A \to 0A0 \mid 1A1 \mid B \\ B \to B0 \mid B1 \mid \# \end{cases}$$

   Notemos que:

   a) La variable S genera Av_2.

   b) La variable A genera $uBu^{-1}v_2$.

   c) La variable B genera $u\#v_1u^{-1}v_2$.

2. $L_2 = \{u_1\#u_2\#\ldots\#u_k \mid k \in \mathbb{N}\backslash\{0\},$ cada $u_i \in \{0,1\}^*,$ y para algún i, j se tiene que $u_i = u_j^{-1}\}$

   En primer lugar, para ayudarnos, usaremos la siguiente regla de producción, que genera cualquier $u_p \in \{0,1\}^*$:

   $$X \to 0X \mid 1X \mid \varepsilon$$

Sean ahora $i, j \in \mathbb{N}$ de forma que $u_i = u_j^{-1}$. Buscamos en primer lugar generar todas las palabras del principio que son anteriores a u_i, es decir, todas las u_p con $p < i$. Esto lo hacemos con la siguiente regla de producción:

$$A \to X\#A \mid \varepsilon$$

De esta forma, tenemos:

$$A \overset{*}{\Longrightarrow} u_1\#u_2\#\ldots\#u_{i-1}\#$$

Ahora, generamos todas las palabras del final que son posteriores a u_j, es decir, todas las u_p con $p > j$. Esto lo hacemos con la siguiente regla de producción:

$$B \to B\#X \mid \varepsilon$$

De esta forma, tenemos:

$$B \overset{*}{\Longrightarrow} \#u_{j+1}\#\ldots\#u_k$$

Ahora, generamos u_i y u_j con la siguiente regla de producción:

$$\begin{array}{ll} P \to 0P0 \mid 1P1 & \text{Parte } u_i = u_j^{-1} \\ P \to 0 \mid 1 \mid \varepsilon & \text{Si } i = j \\ P \to \#A & \text{Si } j \neq i \end{array}$$

Unimos ahora todo en una gramática $G = (V, \{0, 1, \#\}, P, S)$, con $V = \{S, A, B, P, X\}$ y P las reglas de producción siguientes:

$$\begin{array}{l} S \to APB \\ A \to X\#A \mid \varepsilon \\ B \to B\#X \mid \varepsilon \\ P \to 0P0 \mid 1P1 \mid 0 \mid 1 \mid \varepsilon \mid \#A \\ X \to 0X \mid 1X \mid \varepsilon \end{array}$$

Esta gramática es independiente del contexto, y tenemos $L_2 = \mathcal{L}(G)$, luego L_2 es independiente del contexto.

Ejercicio 1.1.17. Sobre el alfabeto $\{0, 1\}$ dar una gramática no ambigua que genere todas las palabras en las que el número de 0s es el doble que el de 1s.

Ejercicio 1.1.18. Sea el lenguaje $L = \{u\#v \mid u, v \in \{0, 1\}^*,\ u \neq v\}$. Demostrar que L es independiente del contexto.

En todos los casos, emplearemos la regla de producción siguiente por comodidad:

$$T \to 0 \mid 1 \quad \text{Símbolo terminal}$$

Sea $n = |u|,\ m = |v|$. La palabra que buscamos generar es:

$$a_1 \cdots a_n\#b_1 \cdots b_m \qquad a_p, b_q \in \{0, 1\}\ \forall p \in \{1, \ldots, n\},\ \forall q \in \{1, \ldots, m\}$$

Distinguimos en función de u, v:

1. Si $a_i = b_i \ \forall i \in \{1, \ldots, \text{mín}\{n, m\}\}$:

   Si tuviésemos $n = m$, entonces $u = v$, por lo que hemos de tener $n \neq m$. Generamos por tanto palabras de la forma $u\#v$ con $n \neq m$. Notemos que todas las palabras de este apartado cumplen efectivamente que $n \neq m$, pero no todas las palabras con $n \neq m$ cumplen la condición de este apartado. Las palabras con $n \neq m$ que no cumplen la condición del apartado se podrán entonces generar con las reglas de producción de este apartado con las del siguiente. Por tanto, la gramática que generaremos es ambigua.

   Buscamos genarar entonces palabras de la forma $u\#v$ con $n \neq m$. En este caso, alguna de las dos palabras tiene más símbolos que la otra, por lo que no hemos de preocuparnos de que sean distintas, ya que esto se tiene al ser de distinta longitud. Estas palabras se generan con:

$A \to TAT$	Parte de la misma longitud
$A \to X_{01}\#$	Fuerza a que $\|u\| > \|v\|$
$A \to \#X_{01}$	Fuerza a que $\|v\| > \|u\|$
$X_{01} \to TX_{01} \mid T$	Genera $z \in \{0,1\}^* \setminus \{\varepsilon\}$

2. Si $\exists i \in \{1, \ldots, \text{mín}\{n, m\}\}$ tal que $a_i \neq b_i$:

   Si el primer símbolo distinto está en la posición i, buscamos generar una palabra de la forma:

   $$\underbrace{a_1 \cdots a_{i-1}}_{3^\circ}\ \underbrace{a_i}_{4^\circ}\ \underbrace{a_{i+1} \cdots a_n}_{5^\circ}\ \underbrace{\#}_{6^\circ}\ \underbrace{b_1 \cdots b_{i-1}}_{3^\circ}\ \underbrace{b_i}_{2^\circ}\ \underbrace{b_{i+1} \cdots b_m}_{1^\circ}$$
   $$a_p, b_q \in \{0,1\}\ \forall p \in \{1, \ldots, n\},\ \forall q \in \{1, \ldots, m\},\ m, n \in \mathbb{N}, a_i \neq b_i$$

   Las reglas de producción son las siguientes:

$C \to CT$	Parte 1°
$C \to C_0 0$	Parte 2°, $b_i = 0$, por lo que $a_i = 1$
$C \to C_1 1$	Parte 2°, $b_i = 1$, por lo que $a_i = 0$
$C_0 \to TC_0T$	Parte 3°, $b_i = 0$, por lo que $a_i = 1$
$C_1 \to TC_1T$	Parte 3°, $b_i = 1$, por lo que $a_i = 0$
$C_0 \to 1X_\#$	Parte 4°, $b_i = 0$, $a_i = 1$
$C_1 \to 0X_\#$	Parte 4°, $b_i = 1$, $a_i = 0$
$X_\# \to TX_\#$	Parte 5°
$X_\# \to \#$	Parte 6°

Por tanto, la gramática que genera L es:

$$G = (V, \{0, 1, \#\}, P, S)$$
$$V = \{S, A, X_{01}, C, C_0, C_1, X_\#\}$$
$$P = \begin{cases} S \to A \mid C \\ T \to 0 \mid 1 \\ A \to TAT \mid X_{01}\# \mid \#X_{01} \\ X_{01} \to TX_{01} \mid T \\ C \to CT \mid C_0 0 \mid C_1 1 \\ C_0 \to TC_0T \mid 1X_\# \\ C_1 \to TC_1T \mid 0X_\# \\ X_\# \to TX_\# \mid \# \end{cases}$$

Ejercicio 1.1.19. Demostrar que si una gramática G está en forma normal de Chomsky, entonces si $w \in \mathcal{L}(G)$ el número de pasos de derivación de toda generación de esta palabra es $2|w| - 1$.

Sabemos que, si G está en forma normal de Chomsky, $\varepsilon \notin \mathcal{L}(G)$. Por tanto, si $w \in \mathcal{L}(G)$, entonces $|w| \geqslant 1$. Sea $|w| = n \in \mathbb{N}$, de forma que:

$$w = a_1 a_2 \cdots a_n, \qquad a_i \in T \ \forall i \in \{1, \ldots, n\}$$

Como la gramática está en forma de Chomsky, todas las producciones son de la siguiente foma:

$$A \to BC \qquad B, C \in V$$
$$A_i \to a_i \qquad a_i \in T$$

De esta forma, por un lado hemos de usar n producciones del segundo tipo para poder llegar así a los símbolos terminales. Además, debemos conseguir las n variables que nos permiten llegar a símbolos iniciales. Sea k el número de veces que empleamos una producción del primer tipo (que son las únicas que nos permiten aumentar en 1). Como cada producción de este tipo aumenta en 1 el número de variables, y usando que partimos con una variable (el símbolo inicial), necesitamos que:

$$1 + k = n \Longrightarrow k = n - 1$$

Por tanto, el número total de pasos de derivación es $2n - 1$, es decir, $2|w| - 1$.

Ejercicio 1.1.20. Dar gramáticas independientes del contexto no ambiguas para los siguientes lenguajes sobre el alfabeto $\{0, 1\}$:

1. El conjunto de palabras w tal que en todo prefijo de w el número de 0s es mayor o igual que el número de 1s.

2. El conjunto de palabras w en las que el número de 0s es mayor o igual que el número de 1s.

Ejercicio 1.1.21. Sea $L = \{0^i 1^j 0^k \mid i \neq j, 2i \neq j\}$. Demostrar que L es independiente del contexto.

Ejercicio 1.1.22. Supongamos la siguiente gramática:

$$G = (V, T, P, \langle \text{SENT} \rangle)$$
$$V = \{\langle \text{SENT} \rangle, \langle \text{IF - THEN} \rangle, \langle \text{IF - THEN - ELSE} \rangle, \langle \text{ASIG} \rangle\}$$
$$T = \{\text{if}, \text{condicion}, \text{then}, \text{else}, a := 1\}$$
$$P = \begin{cases} \langle \text{SENT} \rangle \rightarrow \langle \text{ASIG} \rangle \mid \langle \text{IF - THEN} \rangle \mid \langle \text{IF - THEN - ELSE} \rangle \\ \langle \text{IF - THEN} \rangle \rightarrow \text{if condicion then} \langle \text{SENT} \rangle \\ \langle \text{IF - THEN - ELSE} \rangle \rightarrow \text{if condicion then} \langle \text{SENT} \rangle \text{else} \langle \text{SENT} \rangle \\ \langle \text{ASIG} \rangle \rightarrow \text{a} := 1 \end{cases}$$

1. Demostrar que la gramática es ambigua.

2. Dar una gramática no ambigua que genere el mismo lenguaje.

1.1.1. Preguntas Tipo Test

Se pide discutir la veracidad o falsedad de las siguientes afirmaciones:

1. Si un lenguaje de tipo 2 viene generado por una gramática ambigua, siempre puedo encontrar una gramática no ambigua que genere el mismo lenguaje.

2. En una gramática de tipo 2 ambigua no puede existir una palabra generada con un único árbol de derivación.

3. Dada una gramática independiente del contexto, siempre se puede construir una gramática sin transiciones nulas ni unitarias que genere exactamente el mismo lenguaje que la gramática original.

4. Una gramática independiente del contexto es ambigua si existe una palabra que puede ser generada con dos cadenas de derivación distintas.

5. Un lenguaje inherentemente ambiguo puede ser generado por una gramática ambigua.

6. El lenguaje de las palabras sobre $\{0, 1\}$ con un número impar de ceros es independiente del contexto.

7. Si en una producción de una gramática independiente del contexto, uno de los símbolos que contiene es útil, entonces la producción es útil.

8. Todo árbol de derivación de una palabra en una gramática independiente del contexto está asociado a una única derivación por la izquierda.

9. Para poder aplicar el algoritmo que hemos visto para transformar una gramática a forma normal de Greibach, la gramática tiene que estar en forma normal de Chomsky necesariamente.

10. Sólo hay una derivación por la derecha asociada a un árbol de derivación.

11. Si una gramática independiente del contexto no tiene producciones nulas ni unitarias, entonces si u es una palabra de longitud n generada por la gramática, su derivación se obtiene en un número de pasos no superior a $2n-1$.

12. Cada árbol de derivación de una palabra en una gramática de tipo 2, tiene asociada una única derivación por la izquierda de la misma.

13. Existe un lenguaje con un número finito de palabras que no puede ser generado por una gramática libre de contexto.

14. La gramática compuesta por las reglas de producción $S \to AA$, $A \to aSa$, $A \to a$ no es ambigua.

15. Para poder aplicar el algoritmo que transforma una gramática en forma normal de Greibach es necesario que la gramática esté en forma normal de Chomsky.

16. Un lenguaje libre de contexto es inherentemente ambiguo si existe una gramática ambigua que lo genera.

17. La gramática compuesta por las reglas de producción $S \to A$, $A \to aSa$, $A \to a$ es ambigua.

18. Para generar una palabra de longitud n en una gramática en forma normal de Chomsky hacen falta exactamente $2n-1$ pasos de derivación.

19. Es imposible que una gramática esté en forma normal de Chomsky y Greibach al mismo tiempo.

20. En una gramática independiente del contexto, si una palabra de longitud n es generada, entonces el número de pasos de derivación que se emplean debe de ser menor o igual a $2n-1$.

21. El algoritmo que pasa una gramática a forma normal de Greibach produce siempre el mismo resultado con independencia de cómo se numeren las variables.

22. La gramática compuesta por la siguientes reglas de producción $\{S \to A \mid BA \mid SS,\ B \to a \mid b,\ A \to a\}$ es ambigua.

23. Si una palabra de longitud n es generada por una gramática en forma normal de Greibach, entonces lo es con n pasos de derivación exactamente.

24. En una gramática independiente del contexto puede existir una palabra que es generada con dos derivaciones por la izquierda distintas que tienen el mismo árbol de derivación.

25. Una gramática independiente del contexto genera un lenguaje que puede ser representado por una expresión regular.

26. Para cada autómata finito no determinista M existe una gramática independiente de contexto G tal que $L(M) = L(G)$.

27. Para que un autómata con pila sea determinista es necesario que no tenga transiciones nulas.

28. El algoritmo que pasa una gramatica a forma normal de Greibach produce siempre el mismo resultado con independencia de cómo se numeren las variables.

29. El conjunto de cadenas generado por una gramática independiente del contexto en forma normal de Greibach puede ser reconocido por un autómata finito no determinista con transiciones nulas.

30. La intersección de dos lenguajes regulares da lugar a un lenguaje independiente del contexto.

31. Si L_1 y L_2 son independientes del contexto, no podemos asegurar que $L_1 \cap L_2$ también lo sea.