

Modelos de Computación



Los Del DGIIM, losdeldgiim.github.io

Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada



Esta obra está bajo una Licencia Creative Commons Atribución-NoComercial-SinDerivadas 4.0 Internacional (CC BY-NC-ND 4.0).

Eres libre de compartir y redistribuir el contenido de esta obra en cualquier medio o formato, siempre y cuando des el crédito adecuado a los autores originales y no persigas fines comerciales.

Modelos de Computación

Los Del DGIIM, losdeldgiim.github.io

Arturo Olivares Martos

Granada, 2024-2025

Índice general

1. Relaciones de Problemas	5
1.1. Introducción a la Computación	5
1.1.1. Cálculo de gramáticas	16
1.1.2. Preguntas Tipo Test	36
1.2. Autómatas Finitos	43
1.2.1. Preguntas Tipo Test	67
1.3. Propiedades de los Lenguajes Regulares	76

1. Relaciones de Problemas

1.1. Introducción a la Computación

Ejercicio 1.1.1. Sea la gramática $G = (V, T, P, S)$ dada por:

$$V = \{S, X, Y\}$$

$$T = \{a, b\}$$

$$S = S$$

1. Describe el lenguaje generado por la gramática teniendo en cuenta que P viene descrito por:

$$S \rightarrow XYX$$

$$X \rightarrow aX \mid bX \mid \varepsilon$$

$$Y \rightarrow bbb$$

Sea $L = \{ubbbv \mid u, v \in \{a, b\}^*\}$. Demostraremos mediante doble inclusión que $L = \mathcal{L}(G)$.

- ⊂) Sea $w \in L$. Entonces, $w = ubbbv$ con $u, v \in \{a, b\}^*$. Veamos que $S \xRightarrow{*} w$:

$$S \Longrightarrow XYX \Longrightarrow XbbbX$$

Además, es fácil ver que la regla de producción $X \rightarrow aX \mid bX \mid \varepsilon$ nos permite generar cualquier palabra $u \in \{a, b\}^*$. Por tanto, tenemos que $X \xRightarrow{*} u$ y $X \xRightarrow{*} v$; teniendo así que $S \xRightarrow{*} ubbbv$.

- ⊃) Sea $w \in \mathcal{L}(G)$. Veamos la forma de w :

$$S \Longrightarrow XYX \Longrightarrow XbbbX \Longrightarrow ubbbv \mid u, v \in \{a, b\}^*$$

donde en el último paso hemos empleado lo visto en el apartado anterior de la regla de producción $X \rightarrow aX \mid bX \mid \varepsilon$. Por tanto, $w \in L$.

2. Describe el lenguaje generado por la gramática teniendo en cuenta que P viene descrito por:

$$S \rightarrow aX$$

$$X \rightarrow aX \mid bX \mid \varepsilon$$

Sea $L = \{au \mid u \in \{a, b\}^*\}$. Demostraremos mediante doble inclusión que $L = \mathcal{L}(G)$.

⊂) Sea $w \in L$. Entonces, $w = au$ con $u \in \{a, b\}^*$. Veamos que $S \xRightarrow{*} w$:

$$S \Rightarrow aX \Rightarrow au$$

donde en el último paso hemos empleado lo visto respecto a la regla de producción $X \rightarrow aX \mid bX \mid \varepsilon$. Por tanto, $w \in \mathcal{L}(G)$.

⊃) Sea $w \in \mathcal{L}(G)$. Veamos la forma de w :

$$S \Rightarrow aX \Rightarrow au \mid u \in \{a, b\}^*$$

donde en el último paso hemos empleado lo visto respecto a la regla de producción $X \rightarrow aX \mid bX \mid \varepsilon$. Por tanto, $w \in L$.

3. Describe el lenguaje generado por la gramática teniendo en cuenta que P viene descrito por:

$$\begin{aligned} S &\rightarrow XaXaX \\ X &\rightarrow aX \mid bX \mid \varepsilon \end{aligned}$$

Sea $L = \{uavaw \mid u, v, w \in \{a, b\}^*\}$. Demostraremos mediante doble inclusión que $L = \mathcal{L}(G)$.

⊂) Sea $z \in L$. Entonces, $z = uavaw$ con $u, v, w \in \{a, b\}^*$. Veamos que $S \xRightarrow{*} z$:

$$S \Rightarrow XaXaX \Rightarrow uavaw$$

donde en el último paso hemos empleado lo visto respecto a la regla de producción $X \rightarrow aX \mid bX \mid \varepsilon$. Por tanto, $z \in \mathcal{L}(G)$.

⊃) Sea $z \in \mathcal{L}(G)$. Veamos la forma de z :

$$S \Rightarrow XaXaX \Rightarrow uavaw \mid u, v, w \in \{a, b\}^*$$

donde en el último paso hemos empleado lo visto respecto a la regla de producción $X \rightarrow aX \mid bX \mid \varepsilon$. Por tanto, $z \in L$.

4. Describe el lenguaje generado por la gramática teniendo en cuenta que P viene descrito por:

$$\begin{aligned} S &\rightarrow SS \mid XaXaX \mid \varepsilon \\ X &\rightarrow bX \mid \varepsilon \end{aligned}$$

Sea el lenguaje $L = \{b^i ab^j ab^k \mid i, j, k \in \mathbb{N} \cup \{0\}\}$. Demostraremos mediante doble inclusión que $L^* = \mathcal{L}(G)$.

⊂) Sea $z \in L^* = \bigcup_{i \in \mathbb{N}} L^i$. Sea n el menor número natural tal que $z \in L^n$. Notando por $n_a(z)$ al número de a 's en z , tenemos que $n_a(z) = 2n$. Entonces, $z \in L \cdot \dots \cdot L$ (n veces), por lo que existen $i_1, j_1, k_1, \dots, i_n, j_n, k_n \in \mathbb{N} \cup \{0\}$ tales que $z = b^{i_1} ab^{j_1} ab^{k_1} \cdot \dots \cdot b^{i_n} ab^{j_n} ab^{k_n}$. Veamos que $S \xRightarrow{*} z$:

- Para conseguir el número de a 's deseado, empleamos la regla de producción $S \rightarrow SS$ y reemplazamos una de las S por $XaXaX$. Esto lo hacemos n veces.
 - Posteriormente, cada X la sustituiremos tantas veces como sea necesario por bX para conseguir el número de b 's deseado en cada posición, y finalizaremos con $X \rightarrow \varepsilon$.
- ▷) Sea $z \in \mathcal{L}(G)$, y sea $n_a(z)$ el número de a 's en z . Entonces, como el número de a siempre aumenta de dos en dos, tenemos que $n_a(z) = 2n$ para algún $n \in \mathbb{N} \cup \{0\}$. Veamos la forma de z :
- Para llegar a z , hemos tenido que emplear la regla de producción $S \rightarrow SS \rightarrow SXaXaX$ n veces. Una vez llegados aquí, para eliminar la S (ya que habremos llegado a $n_a(z)$ a 's), empleamos la regla de producción $S \rightarrow \varepsilon$.
 - Posteriormente, para cada X , tan solo podemos emplear la regla de producción $X \rightarrow bX \mid \varepsilon$ para conseguir el número de b 's deseado en cada posición.

Por tanto, es directo ver que $z \in L^n \subseteq L^*$.

Ejercicio 1.1.2. Sea la gramática $G = (V, T, P, S)$. Determinar en cada caso el lenguaje generado por la gramática.

1. Tenga en cuenta que:

$$\begin{aligned}
 V &= \{S, A\} \\
 T &= \{a, b\} \\
 S &= S \\
 P &= \left\{ \begin{array}{lcl} S & \rightarrow & abAS \mid a \\ abA & \rightarrow & baab \\ A & \rightarrow & b \end{array} \right\}
 \end{aligned}$$

Sea $L = \{ua \mid u \in \{abb, baab\}^*\}$. Demostraremos mediante doble inclusión que $L = \mathcal{L}(G)$.

- ▷) Sea $w \in L$. Entonces, $w = ua$ con $u \in \{abb, baab\}^*$. Veamos que $S \xRightarrow{*} w$. Para ello, sabemos que $u \in \{abb, baab\}^* = \bigcup_{i \in \mathbb{N}} \{abb, baab\}^i$. Sea n el menor número natural tal que $u \in \{abb, baab\}^n$, es decir, es una concatenación de n subcadenas, cada una de las cuales es o bien abb o bien $baab$. Veamos que S produce ambas subcadenas:

- Para producir abb , tenemos que $S \rightarrow abAS \rightarrow abbS$.
- Para producir $baab$, tenemos que $S \rightarrow abAS \rightarrow baabS$.

Como vemos, en cada caso podemos concatenar la subcadena necesaria, pero siempre nos quedará una S al final. Usamos la regla de producción $S \rightarrow a$ para eliminarla, llegando así a w , por lo que $S \xRightarrow{*} w$ y $w \in \mathcal{L}(G)$.

- ▷) Sea $w \in \mathcal{L}(G)$. Veamos la forma de w , para lo cual hay dos opciones:

- $S \rightarrow a$: En este caso, habremos finalizado la palabra con a , por lo que habremos añadido la subcadena a a la palabra al final.
- $S \rightarrow abAS$: En este caso, también hay dos opciones:
 - $S \rightarrow abAS \rightarrow baabS$: En este caso, habremos concatenado $baab$ con S , por lo que habremos añadido la subcadena $baab$ a la palabra.
 - $S \rightarrow abAS \rightarrow abbS$: En este caso, habremos concatenado abb con S , por lo que habremos añadido la subcadena abb a la palabra.

Por tanto, w es de la forma ua con u una concatenación de abb 's y $baab$'s, es decir, $u \in \{abb, baab\}^*$. Por tanto, $w \in L$.

2. Tenga en cuenta que:

$$\begin{aligned}
 V &= \{\langle \text{número} \rangle, \langle \text{dígito} \rangle\} \\
 T &= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \\
 S &= \langle \text{número} \rangle \\
 P &= \left\{ \begin{array}{ll} \langle \text{número} \rangle & \rightarrow \langle \text{número} \rangle \langle \text{dígito} \rangle \\ \langle \text{número} \rangle & \rightarrow \langle \text{dígito} \rangle \\ \langle \text{dígito} \rangle & \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{array} \right\}
 \end{aligned}$$

Tenemos que $\mathcal{L}(G)$ es el conjunto de los números naturales, permitiendo tantos ceros a la izquierda como se quiera. Es decir (usando la notación de potencia y concatenación vista para lenguajes):

$$L = \{0^i n \mid i \in \mathbb{N} \cup \{0\}, n \in \mathbb{N} \cup \{0\}\}$$

Demostremoslo mediante doble inclusión que $L = \mathcal{L}(G)$.

⊂) Sea $w \in L$. Entonces, $w = 0^i n$ con $i \in \mathbb{N} \cup \{0\}$ y $n \in \mathbb{N} \cup \{0\}$. Veamos que $\langle \text{número} \rangle \xRightarrow{*} w$:

- En primer lugar, aplicamos $|w| - 1$ veces la regla de producción $\langle \text{número} \rangle \rightarrow \langle \text{número} \rangle \langle \text{dígito} \rangle$ y la regla que lleva de $\langle \text{dígito} \rangle$ a uno de los símbolos terminales, consiguiendo así en cada etapa reemplazar la última variable presente en la cadena por un dígito.
- Finalmente, aplicamos la regla de producción $\langle \text{número} \rangle \rightarrow \langle \text{dígito} \rangle$ para reemplazar la última variable por un dígito, que será el primero del número formado.

Por tanto, $\langle \text{número} \rangle \xRightarrow{*} w$, teniendo que $w \in \mathcal{L}(G)$.

⊃) Sea $w \in \mathcal{L}(G)$. Como la única regla que aumenta la longitud es la regla de producción $\langle \text{número} \rangle \rightarrow \langle \text{número} \rangle \langle \text{dígito} \rangle$, tenemos que w tiene la forma:

$$\begin{aligned}
 \langle \text{número} \rangle &\xRightarrow{*} \langle \text{número} \rangle \langle \text{dígito} \rangle \xRightarrow{|w|-1 \text{ veces}} \\
 &\xRightarrow{*} \langle \text{número} \rangle \langle \text{dígito} \rangle \langle \text{dígito} \rangle \xRightarrow{|w|-1 \text{ veces}} \dots \langle \text{dígito} \rangle \xRightarrow{*} \\
 &\xRightarrow{*} \langle \text{dígito} \rangle \xRightarrow{|w| \text{ veces}} \dots \langle \text{dígito} \rangle
 \end{aligned}$$

Por tanto, tenemos que se trata una sucesión de $|w|$ dígitos, lo que nos lleva a que $w \in L$.

3. Tenga en cuenta que:

$$\begin{aligned} V &= \{A, S\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & aS \mid aA \\ A & \rightarrow & bA \mid b \end{array} \right\} \end{aligned}$$

Sea $L = \{a^n b^m \in \{a, b\}^* \mid n, m \in \mathbb{N}\}$. Demostraremos mediante doble inclusión que $L = \mathcal{L}(G)$.

⊂) Sea $w \in L$. Entonces, $w = a^n b^m$ con $n, m \in \mathbb{N}$. Veamos que $S \xRightarrow{*} w$:

- En primer lugar, aplicamos $n-1$ veces la regla de producción $S \rightarrow aS$ para obtener $a^{n-1}S$,

$$S \xRightarrow{*} a^{n-1}S$$

- Para cambiar a la etapa de añadir b 's, aplicamos la regla de producción $S \rightarrow aA$, obteniendo así $a^n A$,
- Después, aplicamos $m-1$ veces la regla de producción $A \rightarrow bA$ para obtener $a^n b^{m-1} A$.
- Para finalizar, aplicamos la regla de producción $A \rightarrow b$ para obtener $a^n b^m$.

Por tanto, $S \xRightarrow{*} w$, teniendo que $w \in \mathcal{L}(G)$.

⊃) Sea $w \in \mathcal{L}(G)$. Vemos que en la palabra siempre va a haber tan solo una variable (ya sea S o A). Se empezará con la S , y en cierto momento se cambiará a la A , sin poder entonces volver a la S .

- Cuando se está en la etapa en la que hay S , tan solo se pueden añadir a 's, o bien cambiar a la A .
- Cuando se está en la etapa en la que hay A , tan solo se pueden añadir b 's.

Por tanto, tenemos que w estará formada por una sucesión de a 's seguida de una sucesión de b 's, lo que nos lleva a que $w \in L$.

Ejercicio 1.1.3. Encontrar gramáticas de tipo 2 para los siguientes lenguajes sobre el alfabeto $\{a, b\}$. En cada caso determinar si los lenguajes generados son de tipo 3, estudiando si existe una gramática de tipo 3 que los genera.

1. Palabras en las que el número de b no es tres.

Tenemos varias opciones:

- Que no tenga b 's.
- Que tenga una b .
- Que tenga dos b 's.
- Que tenga 4 o más b 's.

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, A, X\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \rightarrow A \mid AbA \mid AbAbA \mid XbXbXbXbX \\ A \rightarrow aA \mid \varepsilon \\ X \rightarrow aX \mid bX \mid \varepsilon \end{array} \right\} \end{aligned}$$

Esta gramática no obstante es de tipo 2. Busquemos otra que sea de tipo 3. Sea la gramática $G' = (V', T', P', S')$ dada por:

$$\begin{aligned} V' &= \{S, X, Y, Z, W\} \\ T' &= \{a, b\} \\ S' &= S \\ P' &= \left\{ \begin{array}{l} S \rightarrow \varepsilon \mid aS \mid bX \\ X \rightarrow \varepsilon \mid aX \mid bY \\ Y \rightarrow \varepsilon \mid aY \mid bZ \\ Z \rightarrow aZ \mid bW \\ W \rightarrow \varepsilon \mid aW \mid bW \end{array} \right\} \end{aligned}$$

Esta sí es de tipo 3, y genera el lenguaje deseado.

2. Palabras que tienen 2 ó 3 b .

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, A, B\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \rightarrow AbAbABA \\ A \rightarrow aA \mid \varepsilon \\ B \rightarrow b \mid \varepsilon \end{array} \right\} \end{aligned}$$

Esta gramática no obstante es de tipo 2. Busquemos otra que sea de tipo 3. Sea la gramática $G' = (V', T', P', S')$ dada por:

$$\begin{aligned} V' &= \{S, X, Y, Z, W, V, T\} \\ T' &= \{a, b\} \\ S' &= S \\ P' &= \left\{ \begin{array}{l} S \rightarrow aS \mid X \\ X \rightarrow bY \\ Y \rightarrow aY \mid Z \\ Z \rightarrow bW \\ W \rightarrow aW \mid \varepsilon \mid V \\ V \rightarrow bT \\ T \rightarrow aT \mid \varepsilon \end{array} \right\} \end{aligned}$$

Esta gramática ya es de tipo 3, pero contiene un número elevado de variables. Veamos si podemos reducirlo: Sea la gramática $G'' = (V'', T'', P'', S'')$ dada por:

$$\begin{aligned} V'' &= \{S, X, Y, Z\} \\ T'' &= \{a, b\} \\ S'' &= S \\ P'' &= \left\{ \begin{array}{lcl} S & \rightarrow & aS \mid bX \\ X & \rightarrow & aX \mid bY \\ Y & \rightarrow & aY \mid \varepsilon \mid bZ \\ Z & \rightarrow & aZ \mid \varepsilon \end{array} \right\} \end{aligned}$$

Notemos que, en esta gramática de tipo 3, ya hemos conseguido el menor número de variables posibles, que representan las 4 etapas. Como la última es opcional, está la regla $Y \rightarrow \varepsilon$, para así no agregar la tercera b .

Ejercicio 1.1.4. Encontrar gramáticas de tipo 2 para los siguientes lenguajes sobre el alfabeto $\{a, b\}$. En cada caso determinar si los lenguajes generados son de tipo 3, estudiando si existe una gramática de tipo 3 que los genera.

1. Palabras que no contienen la subcadena ab .

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, A\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & aA \mid bS \mid \varepsilon \\ A & \rightarrow & aA \mid \varepsilon \end{array} \right\} \end{aligned}$$

Notemos además que esta gramática es de tipo 3, y se tiene que:

$$\mathcal{L}(G) = \{b^i a^j \mid i, j \in \mathbb{N} \cup \{0\}\}$$

2. Palabras que no contienen la subcadena baa .

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, B\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & aS \mid bB \mid \varepsilon \\ B & \rightarrow & bB \mid abB \mid a \mid \varepsilon \end{array} \right\} \end{aligned}$$

Notemos además que esta gramática es de tipo 3.

Ejercicio 1.1.5. Encontrar una gramática libre de contexto que genere el lenguaje sobre el alfabeto $\{a, b\}$ de las palabras que tienen más a que b (al menos una más).

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, S'\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \rightarrow S'aS' \\ S' \rightarrow S'aS' \mid aS'bS' \mid bS'aS' \mid \varepsilon \end{array} \right\} \end{aligned}$$

Ejercicio 1.1.6. Encontrar, si es posible, una gramática regular (o, si no es posible, una gramática libre del contexto) que genere el lenguaje L supuesto que $L \subset \{a, b\}^*$ y verifica:

1. $u \in L$ si, y solamente si, verifica que u no contiene dos símbolos b consecutivos.

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S\} \\ T &= \{a, b\} \\ S &= S \\ P &= \{ S \rightarrow aS \mid baS \mid b \mid \varepsilon \} \end{aligned}$$

2. $u \in L$ si, y solamente si, verifica que u contiene dos símbolos b consecutivos.

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, B, F\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \rightarrow aS \mid bB \\ B \rightarrow bF \mid aS \\ F \rightarrow aF \mid bF \mid \varepsilon \end{array} \right\} \end{aligned}$$

Notemos que, en este caso, tenemos tres estados:

- S : No hemos encontrado dos b 's consecutivas.
- B : Hemos encontrado una b , y puede ser que nos encontremos la segunda b .
- F : Hemos encontrado dos b 's consecutivas; ya hay libertad.

Sí es cierto que usamos tres variables. Para usar solo dos variables, podemos hacer lo siguiente. Sea la gramática $G' = (V', T', P', S')$ dada por:

$$\begin{aligned} V' &= \{S, X\} \\ T' &= \{a, b\} \\ S' &= S \\ P' &= \left\{ \begin{array}{l} S \rightarrow aS \mid bS \mid bbX \\ X \rightarrow aX \mid bX \mid \varepsilon \end{array} \right\} \end{aligned}$$

Ejercicio 1.1.7. Encontrar, si es posible, una gramática regular (o, si no es posible, una gramática libre del contexto) que genere el lenguaje L supuesto que $L \subset \{a, b\}^*$ y verifica:

1. $u \in L$ si, y solamente si, verifica que contiene un número impar de símbolos a .

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \rightarrow aX \mid bS \\ X \rightarrow aS \mid bX \mid \varepsilon \end{array} \right\} \end{aligned}$$

2. $u \in L$ si, y solamente si, verifica que no contiene el mismo número de símbolos a que de símbolos b .

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, A, B, X\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \rightarrow AaA \mid BbB \\ A \rightarrow AaA \mid X \\ B \rightarrow BbB \mid X \\ X \rightarrow aXbX \mid bXaX \mid \varepsilon \end{array} \right\} \end{aligned}$$

Ejercicio 1.1.8. Dado el alfabeto $A = \{a, b\}$ determinar si es posible encontrar una gramática libre de contexto que:

1. Genere las palabras de longitud impar, y mayor o igual que 3, tales que la primera letra coincida con la letra central de la palabra.

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X, A, B, C, D\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \rightarrow A \mid B \\ A \rightarrow aCX \\ C \rightarrow a \mid XCX \\ B \rightarrow bDX \\ D \rightarrow b \mid XDX \\ X \rightarrow a \mid b \end{array} \right\} \end{aligned}$$

2. Genere las palabras de longitud par, y mayor o igual que 2, tales que las dos letras centrales coincidan.

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & XSX \mid C \\ C & \rightarrow & aa \mid bb \\ X & \rightarrow & a \mid b \end{array} \right\} \end{aligned}$$

Ejercicio 1.1.9. Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & SS \\ S & \rightarrow & XXX \\ X & \rightarrow & aX \mid Xa \mid b \end{array} \right\} \end{aligned}$$

Determinar si el lenguaje generado por la gramática es regular. Justificar la respuesta.

Sea la siguiente gramática regular $G' = (V', T', P', S')$ dada por:

$$\begin{aligned} V' &= \{S, X, Y, Z\} \\ T' &= \{a, b\} \\ S' &= S \\ P' &= \left\{ \begin{array}{lcl} S & \rightarrow & aS \mid bX \\ X & \rightarrow & aX \mid bY \\ Y & \rightarrow & aY \mid bZ \\ Z & \rightarrow & aZ \mid bX \mid \varepsilon \end{array} \right\} \end{aligned}$$

Tenemos que $\mathcal{L}(G) = \mathcal{L}(G')$, y como G' es una gramática regular, tenemos que $\mathcal{L}(G)$ es regular. Sí es cierto que en el tema 2 aprendemos otras maneras de demostrarlo más sencillas, como buscar un autómata finito que lo genere.

Ejercicio 1.1.10. Dado un lenguaje L sobre un alfabeto A , ¿es L^* siempre numerable? ¿nunca lo es? ¿o puede serlo unas veces sí y otras, no? Pon ejemplos en este último caso.

L^* es siempre numerable, veámos por qué. L^* es un lenguaje sobre el alfabeto A , por lo que $L^* \subseteq A^*$ y A^* es numerable (visto en teoría), luego L^* también lo es.

Ejercicio 1.1.11. Dado un lenguaje L sobre un alfabeto A , caracterizar cuando $L^* = L$. Esto es, dar un conjunto de propiedades sobre L de manera que L cumpla esas propiedades si y sólo si $L^* = L$.

$$L = L^* \iff \left\{ \begin{array}{l} \varepsilon \in L \\ \wedge \\ u, v \in L \implies uv \in L \end{array} \right.$$

Es decir, $L = L^*$ si y solo si la cadena vacía está en L y además es cerrado para concatenaciones.

Demostración. Demostramos mediante doble implicación.

\Leftarrow) La inclusión $L \subseteq L^*$ es obvia, por lo que solo falta demostrar la otra inclusión.

Sea $v \in L^*$:

1. Si $v = \varepsilon \implies v \in L$ por hipótesis.
2. Si $v \neq \varepsilon$, $\exists n \in \mathbb{N}$ tal que

$$v = a_1 a_2 \dots a_n$$

con $a_i \in L \forall i \in \{1, \dots, n\}$, de donde tenemos que $v \in L$, por ser cerrado para concatenaciones. Luego $L^* \subseteq L$.

\implies) Hemos de probar dos cosas:

1. $\varepsilon \in L^* = L$.
2. Sean $u, v \in L = L^* \implies uv \in L^* = L$.

□

Ejercicio 1.1.12. Dados dos homomorfismos $f : A^* \rightarrow B^*$, $g : A^* \rightarrow B^*$, se dice que son iguales si $f(x) = g(x)$, $\forall x \in A^*$. ¿Existe un procedimiento algorítmico para comprobar si dos homomorfismos son iguales?

Sí, basta probar que su imagen coincide sobre un conjunto finito de elementos, los de A :

$$f(x) = g(x) \quad \forall x \in A^* \iff f(a) = g(a) \quad \forall a \in A$$

Demostración.

\Leftarrow) Sea $v \in A^*$, $\exists n \in \mathbb{N}$ tal que $v = a_1 a_2 \dots a_n$ con $a_i \in A \forall i \in \{1, \dots, n\}$

$$f(v) = f(a_1) f(a_2) \dots f(a_n) = g(a_1) g(a_2) \dots g(a_n) = g(v)$$

\implies) Sea $a \in A \implies a \in A^* \implies f(a) = g(a)$.

□

Ejercicio 1.1.13. Sea $L \subseteq A^*$ un lenguaje arbitrario. Sea $C_0 = L$ y definamos los lenguajes S_i y C_i , para todo $i \geq 1$, por $S_i = C_{i-1}^+$ y $C_i = \overline{S_i}$.

1. ¿Es S_1 siempre, nunca o a veces igual a C_2 ? Justifica la respuesta.
2. Demostrar que $S_2 = C_3$, cualquiera que sea L .

Observación. Demuestra que C_2 es cerrado para la concatenación.

Ejercicio 1.1.14. Demuestra que, para todo alfabeto A , el conjunto de los lenguajes finitos sobre dicho alfabeto es numerable.

Sea $A = \{a_1, a_2, \dots, a_n\}$, con $n \in \mathbb{N}$. Definimos el siguiente conjunto:

$$\Gamma = \{L \subseteq A^* \mid L \text{ es finito}\}$$

Dado un símbolo $z \notin A$, definimos el conjunto $B = \{z\} \cup A$. Sea B^* numerable, y busquemos una inyección de Γ en B^* . Dado un lenguaje $L \in \Gamma$, sea $L = \{l_1, l_2, \dots, l_m\}$, con $m \in \mathbb{N}$ y $l_i \in A^* \forall i \in \{1, \dots, m\}$. Definimos la siguiente función:

$$\begin{aligned} f: \Gamma &\longrightarrow B^* \\ L &\longmapsto z l_1 z l_2 \dots z l_m z \end{aligned}$$

Veamos que f es inyectiva. Sean $L_1, L_2 \in \Gamma$ tales que $f(L_1) = f(L_2)$. Entonces,

$$z l_1 z l_2 \dots z l_k z = z l'_1 z l'_2 \dots z l'_{k'} z$$

Por ser ambas palabras iguales, tenemos que $k = k'$ y $l_i = l'_i \forall i \in \{1, \dots, k\}$, de donde $L_1 = L_2$. Por tanto, f es inyectiva, por lo que Γ es inyectivo con un subconjunto de B^* , que es numerable. Por tanto, Γ es numerable.

1.1.1. Cálculo de gramáticas

Ejercicio 1.1.15 (Complejidad: Sencilla). Calcula, de forma razonada, gramáticas que generen cada uno de los siguientes lenguajes:

1. $\{u \in \{0, 1\}^* \mid |u| \leq 4\}$

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X\} \\ T &= \{0, 1\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \rightarrow XXXX \\ X \rightarrow 0 \mid 1 \mid \varepsilon \end{array} \right\} \end{aligned}$$

No obstante, esta gramática es de tipo 2. Busquemos una de tipo 3. Sea la gramática $G' = (V', T', P', S')$ dada por:

$$\begin{aligned} V' &= \{S, X, Y, Z\} \\ T' &= \{0, 1\} \\ S' &= S \\ P' &= \left\{ \begin{array}{l} S \rightarrow 0X \mid 1X \mid \varepsilon \\ X \rightarrow 0Y \mid 1Y \mid \varepsilon \\ Y \rightarrow 0Z \mid 1Z \mid \varepsilon \\ Z \rightarrow 0 \mid 1 \end{array} \right\} \end{aligned}$$

Tenemos que $\mathcal{L}(G) = \mathcal{L}(G')$, y es igual al lenguaje deseado. Tenemos por tanto que es un lenguaje regular.

2. Palabras con 0's y 1's que no contengan dos 1's consecutivos y que empiecen por un 1 y que terminen por dos 0's.

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X, Y\} \\ T &= \{0, 1\} \\ S &= S \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & 1X00 \\ X & \rightarrow & 0Y \mid \varepsilon \\ Y & \rightarrow & 0Y \mid 1X \mid \varepsilon \end{array} \right\} \end{aligned}$$

Notemos que esta gramática es de tipo 2 debido a la primera regla de producción. Busquemos una de tipo 3. Sea la gramática $G' = (V', T', P', S')$ dada por:

$$\begin{aligned} V' &= \{S, X, Y\} \\ T' &= \{0, 1\} \\ S' &= S \\ P' &= \left\{ \begin{array}{lcl} S & \rightarrow & 1X \\ X & \rightarrow & 0Y \mid F \\ Y & \rightarrow & 0Y \mid 1X \mid F \\ F & \rightarrow & 00 \end{array} \right\} \end{aligned}$$

Tenemos que $\mathcal{L}(G) = \mathcal{L}(G')$, y es igual al lenguaje deseado. Tenemos por tanto que es un lenguaje regular. En esta última gramática, tenemos los siguientes estados:

- S : Es el estado inicial, empezamos con un 1.
- X : Acabamos de escribir un 1, por lo que ahora tan solo podemos escribir 0's.
- Y : Acabamos de escribir un 0, por lo que ahora podemos escribir tanto 0's como 1's.
- F : Ya hemos terminado, y escribimos los dos 0's finales por la restricción impuesta.

3. El conjunto vacío.

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S\} \\ T &= \emptyset \\ S &= S \\ P &= \{ S \rightarrow S \} \end{aligned}$$

4. El lenguaje formado por los números naturales.

Sea la gramática $G = (V, T, P, S)$ dada por:

$$V = \{\langle \text{número no iniciado} \rangle, \langle \text{dígito no cero} \rangle, \langle \text{dígito} \rangle, \langle \text{número iniciado} \rangle\}$$

$$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$S = \langle \text{número no iniciado} \rangle$$

$$P = \left\{ \begin{array}{ll} \langle \text{número no iniciado} \rangle & \rightarrow \langle \text{dígito no cero} \rangle \mid \langle \text{dígito no cero} \rangle \langle \text{número iniciado} \rangle \\ \langle \text{número iniciado} \rangle & \rightarrow \langle \text{dígito} \rangle \mid \langle \text{dígito} \rangle \langle \text{número iniciado} \rangle \\ \langle \text{dígito no cero} \rangle & \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\ \langle \text{dígito} \rangle & \rightarrow 0 \mid \langle \text{dígito no cero} \rangle \end{array} \right\}$$

Notemos que esta gramática es similar a la descrita en el Ejercicio 1.1.2.2, pero adaptada para que los números naturales no puedan empezar por 0. No obstante, esta gramática es de tipo 2. Busquemos una de tipo 3. Sea la gramática $G' = (V', T', P', S')$ dada por:

$$V' = \{S, X, Y, Z\}$$

$$T' = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$S' = S$$

$$P' = \left\{ \begin{array}{ll} S & \rightarrow 0 \mid 1N \mid 2N \mid 3N \mid 4N \mid 5N \mid 6N \mid 7N \mid 8N \mid 9N \\ N & \rightarrow 0N \mid 1N \mid 2N \mid 3N \mid 4N \mid 5N \mid 6N \mid 7N \mid 8N \mid 9N \mid \varepsilon \end{array} \right\}$$

$$5. \{a^n \in \{a, b\}^* \mid n \geq 0\} \cup \{a^n b^n \in \{a, b\}^* \mid n \geq 0\}$$

Sea la gramática $G = (V, T, P, S)$ dada por:

$$V = \{S, X, Y\}$$

$$T = \{a, b\}$$

$$S = S$$

$$P = \left\{ \begin{array}{ll} S & \rightarrow X \mid Y \mid \varepsilon \\ X & \rightarrow aX \mid \varepsilon \\ Y & \rightarrow aYb \mid \varepsilon \end{array} \right\}$$

$$6. \{a^n b^{2n} c^m \in \{a, b, c\}^* \mid n, m > 0\}$$

Sea la gramática $G = (V, T, P, S)$ dada por:

$$V = \{S, X, Y, Z\}$$

$$T = \{a, b, c\}$$

$$S = S$$

$$P = \left\{ \begin{array}{ll} S & \rightarrow aXbbcY \\ X & \rightarrow aXbb \mid \varepsilon \\ Y & \rightarrow cY \mid \varepsilon \end{array} \right\}$$

$$7. \{a^n b^m a^n \in \{a, b\}^* \mid m, n \geq 0\}$$

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \rightarrow aSa \mid bX \mid \varepsilon \\ X \rightarrow bX \mid \varepsilon \end{array} \right\} \end{aligned}$$

8. Palabras con 0's y 1's que contengan la subcadena 00 y 11.

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X\} \\ T &= \{0, 1\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \rightarrow X00X11X \mid X11X00X \\ X \rightarrow 0X \mid 1X \mid \varepsilon \end{array} \right\} \end{aligned}$$

Notemos que esta gramática es de tipo 2. Busquemos una de tipo 3. Sea la gramática $G' = (V', T', P', S')$ dada por:

$$\begin{aligned} V' &= \{S, X, A, B, F\} \\ T' &= \{0, 1\} \\ S' &= S \\ P' &= \left\{ \begin{array}{l} S \rightarrow 0S \mid 1S \mid X \\ X \rightarrow 00A \mid 11B \\ A \rightarrow 0A \mid 1A \mid 11F \\ B \rightarrow 0B \mid 1B \mid 00F \\ F \rightarrow 0F \mid 1F \mid \varepsilon \end{array} \right\} \end{aligned}$$

Notemos que:

- S : No hemos encontrado ninguna subcadena.
- X : Hemos encontrado una subcadena, y ahora buscamos la otra.
- A : Hemos encontrado la subcadena 00, y ahora buscamos la subcadena 11.
- B : Hemos encontrado la subcadena 11, y ahora buscamos la subcadena 00.
- F : Hemos encontrado ambas subcadenas.

9. Palíndromos formados con las letras a y b .

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X, Y\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{l} S \rightarrow aSa \mid bSb \mid \varepsilon \mid a \mid b \end{array} \right\} \end{aligned}$$

Notemos que las reglas $S \rightarrow a \mid b$ se han añadido para añadir los palíndromos de longitud impar.

Ejercicio 1.1.16 (Complejidad: Media). Calcula, de forma razonada, gramáticas que generen cada uno de los siguientes lenguajes:

1. $\{uv \in \{0,1\}^* \mid u^{-1} \text{ es un prefijo de } v\}$

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X, Y\} \\ T &= \{0, 1\} \\ S &= S \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & XY \\ X & \rightarrow & 0X0 \mid 1X1 \mid \varepsilon \\ Y & \rightarrow & 0Y \mid 1Y \mid \varepsilon \end{array} \right\} \end{aligned}$$

Notemos que X deriva en el palíndromo, uu^{-1} , y Y en el resto de la palabra de v .

2. $\{ucv \in \{a,b,c\}^* \mid |u| = |v|\}$

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X\} \\ T &= \{a, b, c\} \\ S &= S \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & XSX \mid c \\ X & \rightarrow & a \mid b \mid c \end{array} \right\} \end{aligned}$$

3. $\{u1^n \in \{0,1\}^* \mid |u| = n\}$

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X\} \\ T &= \{0, 1\} \\ S &= S \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & XS1 \mid \varepsilon \\ X & \rightarrow & 0 \mid 1 \end{array} \right\} \end{aligned}$$

4. $\{a^n b^n a^{n+1} \in \{a,b\}^* \mid n \geq 0\}$ (observar transparencias de teoría)

Sea la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, X, Y\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{lcl} S & \rightarrow & a \mid abaa \mid aXbaa \\ Xb & \rightarrow & bX \\ Xa & \rightarrow & Ybaa \\ bY & \rightarrow & Yb \\ aY & \rightarrow & aa \mid aaX \end{array} \right\} \end{aligned}$$

Ejercicio 1.1.17 (Complejidad: Difícil). Calcula, de forma razonada, gramáticas que generen cada uno de los siguientes lenguajes:

1. $\{a^n b^m c^k \in \{a, b, c\}^* \mid k = m + n\}$

Sea la gramática $G = (V, T, P, S)$ dada por:

$$V = \{S, X\}$$

$$T = \{a, b, c\}$$

$$S = S$$

$$P = \left\{ \begin{array}{l} S \rightarrow aSc \mid X \\ X \rightarrow bXc \mid \varepsilon \end{array} \right\}$$

2. Palabras que son múltiplos de 7 en binario.

Opción 1. Hacer un autómata que acepte el lenguaje. Aunque un concepto del Tema 2, lo añadimos por ser más simple que la segunda opción. Cada estado, donde N es el número que llevamos leído, viene notado por:

$$q_i : N \bmod 7 = i \quad \forall i \in \{0, \dots, 6\}$$

Usamos que:

- Añadirle un 0 al final a un número en binario es multiplicarlo por 2.
- Añadirle un 1 al final a un número en binario es multiplicarlo por 2 y sumarle 1.

Por tanto, el AFD sería:

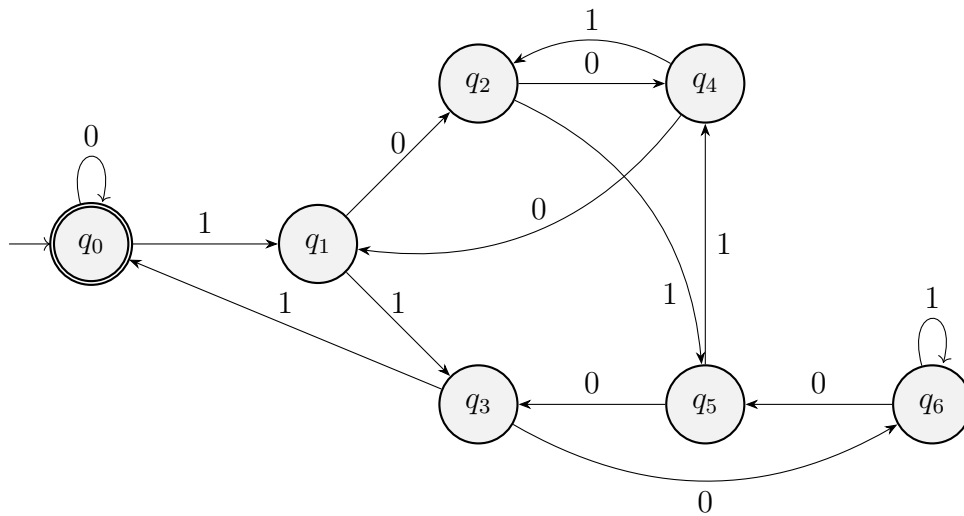


Figura 1.1: AFD que acepta los múltiplos de 7 en binario.

La gramática, por tanto, que genera este lenguaje es $G = (V, T, P, S)$

dada por:

$$V = Q = \{q_0, \dots, q_6\}$$

$$T = A = \{0, 1\}$$

$$S = q_0$$

$$P = \left\{ \begin{array}{lcl} q_0 & \rightarrow & 0q_0 \mid 1q_1 \mid \varepsilon \\ q_1 & \rightarrow & 0q_2 \mid 1q_3 \\ q_2 & \rightarrow & 0q_4 \mid 1q_5 \\ q_3 & \rightarrow & 0q_6 \mid 1q_0 \\ q_4 & \rightarrow & 0q_1 \mid 1q_2 \\ q_5 & \rightarrow & 0q_3 \mid 1q_4 \\ q_6 & \rightarrow & 0q_5 \mid 1q_6 \end{array} \right.$$

Opción 2. Un tanto más complicada, introduce cálculos con números binarios. La idea principal es que si x es un número natural, entonces:

$$7x = (8 - 1)x = 8x - x$$

- Multiplicar un número en binario por 8 es añadirle tres 0s al final.
- Restar un número binario menos otro es realizarle el complemento a dos al segundo, sumar los números y descartar el primer 1.

Realizar estas dos operaciones es más sencillo que multiplicar un número cualquiera en binario por 7. Procedemos por tanto, a:

- a) Generar un número cualquiera en binario.
- b) Multiplicarlo por 8.
- c) Generar su complemento a 2 en binario.
- d) Sumar ambos números.
- e) Descartar el bit de acarreo (el más significativo).

Para esta opción, construiremos la gramática $G = (V, T, P, S)$ dada por:

$$V = \{S, N, \alpha, \beta, \delta, \gamma, Z, Z', A, D, E, E_0, E_1, E_2, E'_0, E'_1, \overline{E_0}, \overline{E_1}, \overline{E_2}, L_0, L_1, X\}$$

$$T = \{0, 1\}$$

$$S = S$$

Y P es un conjunto que contiene todas las reglas de producción que se mostrarán a continuación.

La idea es:

- Generar entre α y β cualquier número en binario, mientras generamos entre β y γ su complemento a 1 en espejo (es decir, el número invertido). Finalmente, multiplicaremos el de la izquierda por 8 y se verá reflejado en la izquierda con 1s.
- Posteriormente, usaremos la variable Z para sumarle 1 al complemento a 1 del número generado.
- Como no podemos modificar símbolos terminales una vez ya generados, trabajaremos todo el rato hasta el final con variables, de forma que A será un 0 y B un 1.

- Una vez generado el número $8x$ y x en complemento a dos en espejo, pasaremos a sumar ambos números usando para ello las variables E y L . Los símbolos del número en complemento a 2 los iremos eliminando y en la izquierda controlaremos los bits del número que ya hemos usado con la variable δ .
- Cuando las variables β y γ “se toquen”, habremos terminado de sumar y ya sólo quedará eliminar las variables delimitadoras (las letras griegas) y sustituir A y B por 0 y 1, respectivamente.

Comenzamos ya describiendo las reglas de producción:

- En primer lugar, creamos el entorno en el que trabajaremos, aceptando “0” como número en binario múltiplo de 7:

$$S \rightarrow \alpha B N A B B B \gamma \mid 0$$

Iremos usando N para generar nuestro número a su izquierda y el complemento a 1 en espejo a la derecha.

- Las tres B s ya introducidas a la derecha son para luego compensar la multiplicación por 8.
- Así mismo, hemos generado ya B a la izquierda y A a la derecha para aceptar sólo números binarios que comiencen por 1.

Usamos ahora la variable N para generar cualquier número en binario a la izquierda, con su complemento a 1 en espejo a la derecha:

$$N \rightarrow A N B \mid B N A \mid A A A \gamma \beta Z$$

Una vez generado el número, terminaremos añadiendo 3 A s a la izquierda (multiplicar por 8), incluyendo los separadores γ y β y la variable Z , que se encargará de sumar 1 al número en complemento a 1 para pasarlo a complemento a 2.

- Usamos ahora Z para pasar el número de la derecha a complemento a 2:

$$Z B \rightarrow B Z$$

Buscamos el primer 0, por lo que saltamos los 1s.

$$Z A \rightarrow Z' B$$

Hemos encontrado el primer 0, lo cambiamos por 1 y volvemos con la variable Z' .

$$B Z' \rightarrow Z' A$$

Volvemos a la izquierda, cambiando todos los 1s que saltamos anteriormente por 0s.

$$\beta Z' \rightarrow \beta L_0$$

Una vez llegamos a β , tenemos el número en complemento a 1 y comenzamos con la aritmética (L_0 representa que no hemos cogido ningún número y que no nos llevamos nada de la suma anterior).

- Comenzamos ahora con la aritmética, la parte más complicada de la gramática. Distinguiamos dos casos:
 - a) No nos llevamos nada de la operación anterior (L_0):

$$L_0A \rightarrow E_0$$

Cogemos un 0 de la derecha y la variable E_0 lo transportará a la izquierda.

$$AE_0 \rightarrow E_0A$$

$$BE_0 \rightarrow E_0B$$

$$\beta E_0 \rightarrow E_0\beta$$

Nos movemos hacia la izquierda, buscando δ (que indica por dónde nos quedamos sumando).

$$\delta E_0 \rightarrow \overline{E_0}$$

Donde la barra indica que hemos “cogido” δ , la cual tendremos que soltar en el siguiente dígito.

$$A\overline{E_0} \rightarrow \delta AE'_0$$

$$B\overline{E_0} \rightarrow \delta BE'_0$$

Como estamos sumando 0, dejamos el dígito invariante, sólo movemos δ hacia la izquierda. Usamos la variable E'_0 para volver, que indica que no nos llevamos nada de la suma:

$$E'_0A \rightarrow AE'_0$$

$$E'_0B \rightarrow BE'_0$$

$$E'_0\beta \rightarrow \beta L_0$$

Nos desplazamos hacia la derecha, hasta encontrar β , ya que después encontraremos el siguiente dígito con el que operar. Como no nos llevábamos nada, volvemos a L_0 .

Si ahora no nos llevamos nada y en vez de un 0 (una A) hay un 1 (una B), repetimos el proceso pero usando para ello E_1 :

$$L_0B \rightarrow E_1$$

$$AE_1 \rightarrow E_1A$$

$$BE_1 \rightarrow E_1B$$

$$\beta E_1 \rightarrow E_1\beta$$

$$\delta E_1 \rightarrow \overline{E_1}$$

A continuación, $\overline{E_1}$ se encontrará con el dígito con el que operar:

$$A\overline{E_1} \rightarrow \delta BE'_0$$

$$B\overline{E_1} \rightarrow \delta AE'_1$$

- Si era un 0 (una A), lo cambiamos por un 1.
- Si era un 1 (una B), lo cambiamos por un 0 y nos llevamos 1 (que es lo que indica E'_1).

El comportamiento de E'_1 es similar a E'_0 pero ahora pasando a L_1 :

$$E'_1 A \rightarrow A E'_1$$

$$E'_1 B \rightarrow B E'_1$$

$$E'_1 \beta \rightarrow \beta L_1$$

- b) Ahora, estamos en el caso en el que nos llevamos un 1 de la operación anterior (L_1), que hemos visto que puede suceder:

$$L_1 A \rightarrow E_1$$

Si nos encontramos un 0 llevando 1, es como si nos hubiéramos encontrado un 1 llevando 0, por lo que no hay nada nuevo que hacer. Sin embargo, si nos encontramos un 1:

$$L_1 B \rightarrow E_2$$

Tenemos que tener en mente que el siguiente dígito con el que realizar la suma lo sumaremos con 2 (E_2 es análogo a E_0 y E_1 pero ahora “transportando” un 2):

$$A E_2 \rightarrow E_2 A$$

$$B E_2 \rightarrow E_2 B$$

$$\beta E_2 \rightarrow E_2 \beta$$

$$\delta E_2 \rightarrow \overline{E_2}$$

A continuación, $\overline{E_2}$ se encontrará con el dígito con el que operar:

$$A \overline{E_2} \rightarrow \delta A E'_1$$

$$B \overline{E_2} \rightarrow \delta B E'_1$$

Similar al caso de $\overline{E_0}$, dejamos el dígito invariante pero ahora tenemos que llevarnos 1 para la siguiente operación.

- A poco que se piense, como $8x$ y su complemento a 2 tienen la misma cantidad de bits, terminaremos de realizar la operación cuando nos llevemos 1 y no queden bits del número en complemento a 2, dando lugar a:

$$\beta L_1 \gamma \rightarrow X$$

donde X es una variable finalizadora, que usamos para cambiar las A s por 0 s, las B s por 1 s y eliminar las variables auxiliares que nos quedan (ya hemos eliminado β y γ directamente al crear X , por lo que nos quedan δ y α):

$$A X \rightarrow X 0$$

$$B X \rightarrow X 1$$

$$\delta X \rightarrow X$$

$$\alpha X \rightarrow \varepsilon$$

Cuando lleguemos a αX , habremos “limpiado” la palabra, por lo que ya podemos quitar todas las variables, generando una palabra de la gramática, que forzosamente tiene que ser un múltiplo de 7 en binario (acabamos de multiplicar cualquier número por 7). Además, como con esta gramática podemos multiplicar cualquier número por 7, esta genera todos los números que son múltiplos de 7 en binario.

Mostramos finalmente un ejemplo de producción de una palabra mediante esta gramática. Trataremos de generar “14” en binario (la 3ª palabra que usa menos reglas de producción para ser creada, tras 0 y 7):

$$\begin{aligned}
S &\rightarrow \alpha BNABBB\gamma \rightarrow \alpha BANBABBB\gamma \rightarrow \alpha BAAAA\delta\beta ZBABBB\gamma \rightarrow \\
&\rightarrow \alpha BAAAA\delta\beta BZABBB\gamma \rightarrow \alpha BAAAA\delta\beta BZ'BBBB\gamma \rightarrow \\
&\rightarrow \alpha BAAAA\delta\beta Z'ABBB\gamma \rightarrow \alpha BAAAA\delta\beta L_0ABBB\gamma \rightarrow \\
&\rightarrow \alpha BAAAA\delta\beta E_0BBBB\gamma \rightarrow \alpha BAAAA\delta E_0\beta BBBB\gamma \rightarrow \\
&\rightarrow \alpha BAAAA\overline{E_0}\beta BBBB\gamma \rightarrow \alpha BAAA\delta AE'_0\beta BBBB\gamma \rightarrow \\
&\rightarrow \alpha BAAA\delta A\beta L_0BBBB\gamma \rightarrow \alpha BAAA\delta A\beta E_1BBB\gamma \rightarrow \\
&\rightarrow \alpha BAAA\delta AE_1\beta BBB\gamma \rightarrow \alpha BAAA\delta E_1A\beta BBB\gamma \rightarrow \\
&\rightarrow \alpha BAAA\overline{E_1}A\beta BBB\gamma \rightarrow \alpha BAA\delta BE'_0A\beta BBB\gamma \rightarrow \\
&\rightarrow \alpha BAA\delta BAE'_0\beta BBB\gamma \rightarrow \alpha BAA\delta BA\beta L_0BBB\gamma \rightarrow \\
&\rightarrow \alpha BAA\delta BA\beta E_1BB\gamma \rightarrow \alpha BAA\delta BAE_1\beta BB\gamma \rightarrow \\
&\rightarrow \alpha BAA\delta BE_1A\beta BB\gamma \rightarrow \alpha BAA\delta E_1BA\beta BB\gamma \rightarrow \\
&\rightarrow \alpha BAA\overline{E_1}BA\beta BB\gamma \rightarrow \alpha BA\delta BE'_0BA\beta BB\gamma \rightarrow \\
&\rightarrow \alpha BA\delta BBE'_0A\beta BB\gamma \rightarrow \alpha BA\delta BB AE'_0\beta BB\gamma \rightarrow \\
&\rightarrow \alpha BA\delta BB A\beta L_0BB\gamma \rightarrow \alpha BA\delta BB A\beta E_1B\gamma \rightarrow \alpha BA\delta BB AE_1\beta B\gamma \rightarrow \\
&\rightarrow \alpha BA\delta BB E_1A\beta B\gamma \rightarrow \alpha BA\delta BE_1BA\beta B\gamma \rightarrow \alpha BA\delta E_1BBA\beta B\gamma \rightarrow \\
&\rightarrow \alpha BA\overline{E_1}BBA\beta B\gamma \rightarrow \alpha B\delta BE'_0BBA\beta B\gamma \rightarrow \alpha B\delta BBE'_0BA\beta B\gamma \rightarrow \\
&\rightarrow \alpha B\delta BB BE'_0A\beta B\gamma \rightarrow \alpha B\delta BB BAE'_0\beta B\gamma \rightarrow \alpha B\delta BB B A\beta L_0B\gamma \rightarrow \\
&\rightarrow \alpha B\delta BB B A\beta E_1\gamma \rightarrow \alpha B\delta BB B AE_1\beta \gamma \rightarrow \alpha B\delta BB BE_1A\beta \gamma \rightarrow \\
&\rightarrow \alpha B\delta BB E_1BA\beta \gamma \rightarrow \alpha B\delta BE_1BBA\beta \gamma \rightarrow \alpha B\delta E_1BBBA\delta \gamma \rightarrow \\
&\rightarrow \alpha B\overline{E_1}BBBA\beta \gamma \rightarrow \alpha \delta AE'_1BBBA\beta \gamma \rightarrow \alpha \delta ABE'_1BBA\beta \gamma \rightarrow \\
&\rightarrow \alpha \delta AB BE'_1BA\beta \gamma \rightarrow \alpha \delta AB BB E'_1A\beta \gamma \rightarrow \alpha \delta AB BB AE'_1\beta \gamma \rightarrow \\
&\rightarrow \alpha \delta AB BB A\beta L_1\gamma \rightarrow \alpha \delta AB BB AX \rightarrow \alpha \delta AB BB X0 \rightarrow \alpha \delta AB BX10 \rightarrow \\
&\rightarrow \alpha \delta AB X110 \rightarrow \alpha \delta AX1110 \rightarrow \alpha \delta X01110 \rightarrow \alpha X01110 \rightarrow 01110
\end{aligned}$$

Ejercicio 1.1.18 (Complejidad: Extrema (no son libres de contexto)). Calcula, de forma razonada, gramáticas que generen cada uno de los siguientes lenguajes:

1. $\{ww \mid w \in \{0,1\}^*\}$

Para este lenguaje, hemos construido la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned}
V &= \{S, \alpha, \beta, \gamma, X, E, E_1, E_0, E', B\} \\
T &= \{0, 1\} \\
S &= S
\end{aligned}$$

P que contiene las reglas de producción que se mostrarán a continuación.

La idea principal en la gramática es generar entre las variables α y β cualquier palabra del lenguaje $\{0, 1\}^*$. Posteriormente, iremos copiando dicha palabra a la derecha de β usando para ello las variables E y γ , de forma que con γ controlaremos la parte de la palabra de la izquierda que ya hayamos copiado a la derecha de β .

Finalmente, usaremos B para eliminar cualquier rastro de las variable auxiliares. De esta forma, las reglas de P son:

- Para generar cualquier palabra entre α y β :

$$\begin{aligned} S &\rightarrow \alpha X \beta \\ X &\rightarrow 0X \mid 1X \mid E\gamma \end{aligned}$$

- Para coger un 1 y copiarlo a la derecha:

Hemos de estar al final de la parte de la palabra no copiada (luego ha de ser $x E \gamma$ siendo x 0 o 1). Posteriormente, avanzamos γ a la izquierda para indicar que dicho 1 ya está copiado y cambiamos a la variable que transporta el 1 a la derecha:

$$1 E \gamma \rightarrow \gamma 1 E_1$$

Posteriormente, movemos dicha variable a la derecha:

$$\begin{aligned} E_1 1 &\rightarrow 1 E_1 \\ E_1 0 &\rightarrow 0 E_1 \end{aligned}$$

Cuando lleguemos al final de la palabra de la izquierda, soltamos el 1 al inicio de la palabra de la derecha:

$$E_1 \beta \rightarrow E' \beta 1$$

- Para coger un 0 y copiarlo a la derecha, es una situación análoga pero usamos otra variable:

$$0 E \gamma \rightarrow \gamma 0 E_0$$

$$\begin{aligned} E_0 1 &\rightarrow 1 E_0 \\ E_0 0 &\rightarrow 0 E_0 \end{aligned}$$

$$E_0 \beta \rightarrow E' \beta 0$$

- Ahora, explicamos E' , cuya única funcionalidad es volver al final de la parte no copiada de la palabra de la izquierda:

$$\begin{aligned} 1 E' &\rightarrow E' 1 \\ 0 E' &\rightarrow E' 0 \\ \gamma E' &\rightarrow E \gamma \end{aligned}$$

- La copia de la palabra terminará cuando se de $\alpha E\gamma$ (ya que estará toda la palabra copiada a la derecha). En dicho caso, eliminamos todas las variables auxiliares restantes:

$$\begin{aligned}\alpha E\gamma &\rightarrow B \\ B1 &\rightarrow 1B \\ B0 &\rightarrow 0B \\ B\beta &\rightarrow \varepsilon\end{aligned}$$

Puede demostrarse que el lenguaje generado por esta gramática es el solicitado. Por la complejidad de la gramática, nos limitamos a mostrar un ejemplo para ver de forma intuitiva el buen funcionamiento de la misma.

Trataremos de generar la cadena: 10111011 (es decir, $(1011)^2$):

$$\begin{aligned}S &\rightarrow \alpha X\beta \rightarrow \alpha 1X\beta \rightarrow \alpha 10X\beta \rightarrow \alpha 101X\beta \rightarrow \alpha 1011X\beta \rightarrow \alpha 1011E\gamma\beta \rightarrow \\ &\rightarrow \alpha 101\gamma 1E_1\beta \rightarrow \alpha 101\gamma 1E'\beta 1 \rightarrow \alpha 101\gamma E'1\beta 1 \rightarrow \alpha 101E\gamma 1\beta 1 \rightarrow \\ &\rightarrow \alpha 10\gamma 1E_11\beta 1 \rightarrow \alpha 10\gamma 11E_1\beta 1 \rightarrow \alpha 10\gamma 11E'\beta 11 \rightarrow \alpha 10\gamma 1E'1\beta 11 \rightarrow \\ &\rightarrow \alpha 10\gamma E'11\beta 11 \rightarrow \alpha 10E\gamma 11\beta 11 \rightarrow \alpha 1\gamma 0E_011\beta 11 \rightarrow \alpha 1\gamma 01E_01\beta 11 \rightarrow \\ &\rightarrow \alpha 1\gamma 011E_0\beta 11 \rightarrow \alpha 1\gamma 011E'\beta 011 \rightarrow \alpha 1\gamma 01E'1\beta 011 \rightarrow \alpha 1\gamma 0E'11\beta 011 \rightarrow \\ &\rightarrow \alpha 1\gamma E'011\beta 011 \rightarrow \alpha 1E\gamma 011\beta 011 \rightarrow \alpha \gamma 1E_1011\beta 011 \rightarrow \alpha \gamma 10E_111\beta 011 \rightarrow \\ &\rightarrow \alpha \gamma 101E_11\beta 011 \rightarrow \alpha \gamma 1011E_1\beta 011 \rightarrow \alpha \gamma 1011E'\beta 1011 \rightarrow \alpha \gamma 101E'1\beta 1011 \rightarrow \\ &\rightarrow \alpha \gamma 10E'11\beta 1011 \rightarrow \alpha \gamma 1E'011\beta 1011 \rightarrow \alpha \gamma E'1011\beta 1011 \rightarrow \alpha E\gamma 1011\beta 1011 \rightarrow \\ &\rightarrow B1011\beta 1011 \rightarrow 1B011\beta 1011 \rightarrow 10B11\beta 1011 \rightarrow 101B1\beta 1011 \rightarrow \\ &\rightarrow 1011B\beta 1011 \rightarrow 10111011\end{aligned}$$

2. $\{a^{n^2} \in \{a\}^* \mid n \geq 0\}$

La idea que hemos tenido para hacer una gramática que acepte el lenguaje es la siguiente. Si representamos las 5 primeras palabras del lenguaje (ordenándolas por su longitud):

$$\begin{aligned}\varepsilon \\ a \\ aaaa \\ aaaaaaaaaa \\ aaaaaaaaaaaaaaaaaa\end{aligned}$$

Notemos que podemos ordenar las letras de la siguiente forma (olvidándonos de ε , que no será relevante):

$$\begin{aligned}a \\ aa \quad aa \\ aaa \quad aaa \quad aaa \\ aaaa \quad aaaa \quad aaaa \quad aaaa\end{aligned}$$

De forma que tenemos 1 grupo de 1 “a”, dos grupos de 2 “a”, 3 grupos de 3 “a”, ... Notemos que dados n grupos de n “a”, será sencillo construir $n + 1$ grupos de $n + 1$ “a”, ya que nos bastará con añadir una “a” a cada grupo y con duplicar el último grupo de “a”.

Hemos construido una gramática $G = (V, T, S, P)$ que simula este comportamiento inductivo del lenguaje, con lo que el lenguaje generado por la misma es el solicitado. Tenemos:

$$\begin{aligned} V &= \{\alpha, \beta, \delta, \gamma, \sigma, X, A, E, E_\sigma, \bar{E}, E', I, R, L, Z\} \\ T &= \{0, 1\} \\ S &= S \end{aligned}$$

Donde P es el conjunto de reglas de producción que contiene todas las reglas que explicaremos a continuación.

La idea es que si queremos generar la palabra a^{n^2} , que generemos $n - 1$ As entre α y β . Tendremos ya creada una letra a y lo que haremos será que por cada A que hayamos generado, repitamos el proceso inductivo descrito anteriormente. Además, separaremos los “grupos” de “a” con variables I . Finalmente, para duplicar un grupo de “a”, usaremos las variables δ y σ .

Empezamos generando nuestro entorno en el que trabajaremos (o la palabra vacía):

$$S \rightarrow \alpha X \beta I a \delta \gamma \mid \varepsilon$$

A continuación, usamos X para generar las As:

$$X \rightarrow AX \mid E$$

Una vez terminadas de leer las As, generaremos E , que se encargará de ir eliminando una A , de realizar el proceso inductivo y de volver al estado inicial, hasta terminar con todas las As generadas.

Ahora, hacemos que E coja una A , con lo que le dejamos salir de la región comprendida por α y β :

$$AE\beta \rightarrow \beta E$$

Ahora desplazamos la variable E a la derecha, haciendo que cada vez que entre en un grupo de “a” (cuando pase una variable I) añada una nueva:

$$\begin{aligned} Ea &\rightarrow aE \\ EI &\rightarrow IaE \end{aligned}$$

Cuando la variable E se encuentre con δ , habremos terminado de incrementar las “a”, con lo que tendremos que duplicar ahora el último grupo de “a”. Para ello, prepararemos un entorno, de forma que entre I y σ vayamos generando el nuevo grupo de “a”, entre σ y δ se encuentren las “a” por copiar; y que entre δ y γ se encuentren las “a” que ya hayan sido duplicadas.

De esta forma, cuando E se encuentre con δ , pasaremos a una variable que busque I para colocar delante suya σ :

$$\begin{aligned} E\delta &\rightarrow E_\sigma\delta \\ aE_\sigma &\rightarrow E_\sigma a \\ IE_\sigma &\rightarrow I\sigma R \end{aligned}$$

Ahora, usaremos R para movernos a la derecha tras copiar una letra y L para movernos a la izquierda con el fin de pegar una letra.

$$Ra \rightarrow aR$$

Nos movemos a la derecha

$$aR\delta \rightarrow L\delta a$$

Cuando lleguemos a δ , guardamos una “a” más como copiada.

$$aL \rightarrow La$$

Nos moveremos hacia la izquierda buscando σ para crear una a :

$$\sigma L \rightarrow a\sigma R$$

Pegaremos una a tras σ y repetiremos el proceso.

El proceso terminará cuando no haya más “a” entre σ y δ :

$$\sigma R\delta \rightarrow I\overline{E}$$

Cuando hayamos terminado, colocamos una I para hacer efectivo el nuevo grupo. Finalmente, debemos colocar nuevamente δ a la izquierda de γ para la siguiente vez que copiemos. Usamos para ello \overline{E} :

$$\overline{E}a \rightarrow a\overline{E}$$

Nos movemos a la izquierda buscando γ y cuando la encontremos, colocamos δ :

$$\overline{E}\gamma \rightarrow E'\delta\gamma$$

Usaremos finalmente E' para desplazarnos a la izquierda, tras β , donde volveremos a la variable E , que reiniciará el proceso descrito para realizarlo nuevamente:

$$\begin{aligned} aE' &\rightarrow E'a \\ IE' &\rightarrow E'I \\ \beta E' &\rightarrow E\beta \end{aligned}$$

Este proceso terminará cuando no queden A s por copiar. En dicho caso, pasaremos a una variable Z que eliminará todas las variables auxiliares:

$$\alpha E \rightarrow Z$$

De esta forma, Z se mueve a la derecha, eliminando todas las variables y pasando a través de las letras:

$$\begin{aligned} Z\beta &\rightarrow Z \\ ZI &\rightarrow Z \\ Za &\rightarrow aZ \\ Z\delta &\rightarrow Z \\ Z\gamma &\rightarrow \varepsilon \end{aligned}$$

Como ejemplo y para comprobar que el lenguaje generado por dicha gramática funciona es el deseado mostramos el siguiente ejemplo, en el que generamos a^9 :

$$\begin{aligned} S &\rightarrow \alpha X\beta Ia\delta\gamma \rightarrow \alpha AX\beta Ia\delta\gamma \rightarrow \alpha AAX\beta Ia\delta\gamma \rightarrow \alpha AAE\beta Ia\delta\gamma \rightarrow \alpha A\beta EIa\delta\gamma \rightarrow \\ &\rightarrow \alpha A\beta IaEa\delta\gamma \rightarrow \alpha A\beta IaaE\delta\gamma \rightarrow \alpha A\beta IaaE_\sigma\delta\gamma \rightarrow \alpha A\beta IaE_\sigma a\delta\gamma \rightarrow \\ &\rightarrow \alpha A\beta IE_\sigma aa\delta\gamma \rightarrow \alpha A\beta I\sigma Raad\gamma \rightarrow \alpha A\beta I\sigma aRa\delta\gamma \rightarrow \alpha A\beta I\sigma aaR\delta\gamma \rightarrow \\ &\rightarrow \alpha A\beta I\sigma aL\delta a\gamma \rightarrow \alpha A\beta I\sigma La\delta a\gamma \rightarrow \alpha A\beta Ia\sigma Ra\delta a\gamma \rightarrow \alpha A\beta Ia\sigma aR\delta a\gamma \rightarrow \\ &\rightarrow \alpha A\beta Ia\sigma L\delta aa\gamma \rightarrow \alpha A\beta Iaa\sigma R\delta aa\gamma \rightarrow \alpha A\beta IaaI\bar{E}aa\gamma \rightarrow \alpha A\beta IaaIa\bar{E}a\gamma \rightarrow \\ &\rightarrow \alpha A\beta IaaIaa\bar{E}\gamma \rightarrow \alpha A\beta IaaIaaE'\delta\gamma \rightarrow \alpha A\beta IaaIaE'a\delta\gamma \rightarrow \alpha A\beta IaaIE'aa\delta\gamma \rightarrow \\ &\rightarrow \alpha A\beta IaaE'Iaa\delta\gamma \rightarrow \alpha A\beta IaE'aIaa\delta\gamma \rightarrow \alpha A\beta IE'aaIaa\delta\gamma \rightarrow \\ &\rightarrow \alpha A\beta E'IaaIaa\delta\gamma \rightarrow \alpha AE\beta IaaIaa\delta\gamma \rightarrow \alpha \beta EIaaIaa\delta\gamma \rightarrow \\ &\rightarrow \alpha \beta IaEaaIaa\delta\gamma \rightarrow \alpha \beta IaaEaIaa\delta\gamma \rightarrow \alpha \beta IaaaEIaa\delta\gamma \rightarrow \\ &\rightarrow \alpha \beta IaaaIaEaa\delta\gamma \rightarrow \alpha \beta IAAAIaaEa\delta\gamma \rightarrow \alpha \beta IaaaIaaaE\delta\gamma \rightarrow \\ &\rightarrow \alpha \beta IaaaIaaaE_\sigma\delta\gamma \rightarrow \alpha \beta IaaaIaaE_\sigma a\delta\gamma \rightarrow \alpha \beta IaaaIaE_\sigma aa\delta\gamma \rightarrow \\ &\rightarrow \alpha \beta IaaaIE_\sigma aa\delta\gamma \rightarrow \alpha \beta IaaaI\sigma Raaa\delta\gamma \rightarrow \alpha \beta IaaaI\sigma aRa\delta\gamma \rightarrow \\ &\rightarrow \alpha \beta IaaaI\sigma aaRa\delta\gamma \rightarrow \alpha \beta IaaaI\sigma aaaR\delta\gamma \rightarrow \alpha \beta IaaaI\sigma aaL\delta a\gamma \rightarrow \\ &\rightarrow \alpha \beta IaaaI\sigma aL\delta a\gamma \rightarrow \alpha \beta IaaaI\sigma La\delta a\gamma \rightarrow \alpha \beta IaaaIa\sigma Ra\delta a\gamma \rightarrow \\ &\rightarrow \alpha \beta IaaaIa\sigma aRa\delta a\gamma \rightarrow \alpha \beta IaaaIa\sigma aaR\delta a\gamma \rightarrow \alpha \beta IaaaIa\sigma aL\delta aa\gamma \rightarrow \\ &\rightarrow \alpha \beta IaaaIa\sigma La\delta aa\gamma \rightarrow \alpha \beta IaaaIaa\sigma Ra\delta aa\gamma \rightarrow \alpha \beta IaaaIaa\sigma aR\delta aa\gamma \rightarrow \\ &\rightarrow \alpha \beta IaaaIaa\sigma L\delta aa\gamma \rightarrow \alpha \beta IaaaIaaa\sigma R\delta aa\gamma \rightarrow \alpha \beta IaaaIaaaI\bar{E}aaa\gamma \rightarrow \\ &\rightarrow \alpha \beta IaaaIaaaIa\bar{E}aa\gamma \rightarrow \alpha \beta IaaaIaaaIaa\bar{E}a\gamma \rightarrow \alpha \beta IaaaIaaaIaaa\bar{E}\gamma \rightarrow \\ &\rightarrow \alpha \beta IaaaIaaaIaaaE'\delta\gamma \xrightarrow{(*)} \alpha \beta E'IaaaIaaaIaaa\delta\gamma \rightarrow \alpha E\beta IaaaIaaaIaaa\delta\gamma \rightarrow \\ &\rightarrow Z\beta IaaaIaaaIaaa\delta\gamma \rightarrow ZIaaaIaaaIaaa\delta\gamma \rightarrow ZaaaIaaaIaaa\delta\gamma \rightarrow \\ &\rightarrow aZaaIaaaIaaa\delta\gamma \rightarrow aaZaIaaaIaaa\delta\gamma \rightarrow aaaZIaaaIaaa\delta\gamma \rightarrow \\ &\rightarrow aaaZaaaIaaa\delta\gamma \xrightarrow{(**)} aaaaaaaaaZ\delta\gamma \rightarrow aaaaaaaaaZ\gamma \rightarrow aaaaaaaaa \end{aligned}$$

- Donde en (*) hemos aplicado reiteradas veces que $aE' \rightarrow E'a$ y que $IE' \rightarrow E'I$.
- Donde en (**) hemos aplicado varias veces que $ZI \rightarrow Z$ y que $Za \rightarrow aZ$.

3. $\{a^p \in \{a\}^* \mid p \text{ es primo}\}$

Se subirá próximamente una gramática.

4. $\{a^n b^m \in \{a, b\}^* \mid n \leq m^2\}$

Una vez conocida una gramática para el lenguaje $\{a^{n^2} \mid n \in \mathbb{N}\}$, dar una gramática para este lenguaje es sencillo. Lo que haremos será generar primero un número arbitrarios de A s (variables) y de b s. Seguidamente, generaremos tantas B s como número de b al cuadrado haya (usando para ello la gramática del lenguaje de los cuadrados perfectos), y finalmente, por cada B que tengamos sustituiremos una A por una a , de forma que si se nos gastan las B s y no hemos sustituido todas las A s, entonces era porque teníamos más a s de las permitidas en este lenguaje ($n > m$), con lo que no podremos generar ninguna palabra. Si por el contrario sustituimos todas las A s y nos siguen quedando B s, eliminaremos todas las B s para poder dar una palabra formada solo por símbolos terminales.

Antes de consultar la gramática, recomendamos encarecidamente entender primero la gramática de los cuadrados perfectos, ya que es más sencilla y la usaremos con soltura para dar esta gramática.

Los pasos que hace esta gramática son:

- a) Primero, genera un número indeterminado de A s, tras la variable λ .
- b) Posteriormente, plantea el “entorno” de variables usado en la gramática de los cuadrados perfectos, que parte del caso base de tener una b .
- c) Entre α y β se generan todas las b que tendrá la posibel palabra a generar (menos una, que se generó en el caso base).
- d) Entre β y γ , iremos colocando tantas B s como número de b s al cuadrado haya.
- e) Para controlar que una b ya le hemos usado para generar el cuadrado, en vez de borrarla como hacíamos en la gramática de cuadrados perfectos, la metemos entre el espacio comprendido entre φ y β . De esta forma, el entorno de trabajo será similar a:

$$\lambda A \dots A b \alpha b \dots b E \varphi b \dots b \beta I B \dots B I B \dots B I \dots I B \dots B \delta \gamma$$

- f) Una vez generadas todas las B s, usamos la variable Z para borrar todas las variables auxiliares para generar las B s, dejando la palabra de la forma

$$\lambda A \dots A b \dots b \beta B \dots B H \gamma$$

de forma que el número de B s coincide con el cuadrado del número de b s.

- g) Posteriormente, usaremos la variable H para borrar una B y posteriormente sustituir una A por una a , hasta que se nos acaben las A s (en cuyo caso la palabra es válida y eliminamos todas las variables dejando sólo los símbolos terminales) o las B s (en cuyo caso, había más A s, con lo que no generamos la palabra).

De esta forma, damos la gramática $G = (V, T, P, S)$ dada por:

$$\begin{aligned} V &= \{S, \lambda, \alpha, \beta, \varphi, \delta, \gamma, \sigma, \psi, G, A, B, X, I, E, E_\sigma, E', \bar{E}, L, R, Z, H, H', \bar{H}\} \\ T &= \{a, b\} \end{aligned}$$

Y P el conjunto de todas las producciones explicadas a continuación.

En primer lugar, aceptamos que ε es una palabra del lenguaje. Habiendo considerado dicho caso, comenzamos pues generando todas las As que queramos al inicio de la palabra:

$$\begin{aligned} S &\rightarrow \lambda G \mid \varepsilon \\ G &\rightarrow AG \end{aligned}$$

Cuando hayamos generado todas las As deseadas, situaremos nuestro entorno de trabajo para generar bs y posteriormente crear tantas Bs como el cuadrado del número de bs :

$$G \rightarrow b\alpha X\varphi\beta IB\delta\gamma$$

Y generaremos tantas bs como queramos con la variable X (notemos que ya hemos generado una b y una B , el caso base cuando $n = 1$).

$$X \rightarrow bX \mid E$$

Cuando hayamos ya generado todas las bs , pasamos a generar las Bs , usando para ello la variable E , que realizará un comportamiento iterativo, de forma que coja una b (la sitúe detrás de φ), añada una B en cada subgrupo de Bs (separados por I), que duplique este último subgrupo (utilizando para ello σ , L y R), y que vuelva con E' a donde empezó, delante de φ :

a) En primer lugar, la variable E coge una b (la sitúa tras φ):

$$bE\varphi \rightarrow \varphi bE$$

Y se desplaza a la derecha de φ .

b) Posteriormente, se desplaza a la derecha, añadiendo una B en cada subgrupo de Bs , proceso que terminará cuando se encuentre con δ :

$$\begin{aligned} Eb &\rightarrow bE \\ E\beta &\rightarrow \beta E \\ EI &\rightarrow IBE \quad (\text{Añade una } B) \\ EB &\rightarrow BE \\ E\delta &\rightarrow E_\sigma\delta \end{aligned}$$

c) Una vez topada con δ , pasaremos a realizar la copia del último grupo de Bs , con lo que tendremos que fijar el σ que usábamos en la gramática de los cuadrados perfectos. Para ello:

$$\begin{aligned} BE_\sigma &\rightarrow E_\sigma B \\ IE_\sigma &\rightarrow I\sigma R \end{aligned}$$

- d) Una vez colocado el σ , comienza la copia de las B s, usando para ello las variables L , R y el delimitador δ , que marca las B s que ya han sido copiadas:

$$\begin{aligned} RB &\rightarrow BR \\ BR\delta &\rightarrow L\delta B \quad (\text{Coge una } B) \\ BL &\rightarrow LB \\ \sigma L &\rightarrow B\sigma R \quad (\text{Deja la } B) \end{aligned}$$

- e) El proceso terminará cuando no haya más B s entre σ y δ una vez copiada la última B (con lo que tendremos la variable R). En dicho caso, colocamos la nueva I que delimita la separación con el grupo de B s recién creado y usamos \bar{E} para devolver δ al final del último grupo de B s:

$$\begin{aligned} \sigma R\delta &\rightarrow I\bar{E} \\ \bar{E}B &\rightarrow B\bar{E} \\ \bar{E}\gamma &\rightarrow E'\delta\gamma \end{aligned}$$

- f) Una vez colocada δ , volveremos con E' hacia la izquierda hasta la situación inicial de E , que es tras φ :

$$\begin{aligned} BE' &\rightarrow E'B \\ IE' &\rightarrow E'I \\ \beta E' &\rightarrow E'\beta \\ bE' &\rightarrow E'b \\ \varphi E' &\rightarrow E\varphi \end{aligned}$$

Como hemos mencionado anteriormente, la variable E repetirá este proceso, hasta quedarse sin bs por realizar su cuadrado. Hasta este punto, la palabra generada será similar a:

$$\lambda A \dots Ab \alpha E \varphi b \dots b \beta I B \dots B I B \dots B I \dots I B \dots B \delta \gamma$$

donde tenemos tantas B s como número de bs al cuadrado (gracias al funcionamiento de la gramática de los cuadrados perfectos). A continuación, usaremos la variable Z para borrar las variables que ya no nos hacen falta, como α , φ , β , I y δ :

$$\begin{aligned} \alpha E \varphi &\rightarrow Z \quad (\text{No quedan } bs \text{ por copiar}) \\ Zb &\rightarrow bZ \\ Z\beta &\rightarrow \beta Z \\ ZI &\rightarrow Z \\ ZB &\rightarrow BZ \\ Z\delta &\rightarrow H \end{aligned}$$

Ahora, usaremos H para cambiar una A por una a por cada B que borremos:

a) En primer lugar, borramos una B :

$$BBH \rightarrow H'B \quad (\text{No es la última})$$

b) A continuación, nos desplazamos hacia la izquierda, buscando la primera A :

$$\begin{aligned} BH' &\rightarrow H'B \\ \beta H' &\rightarrow H'\beta \\ bH' &\rightarrow H'b \\ aH' &\rightarrow H'a \quad (\text{Puede que ya hayamos sustituido alguna}) \end{aligned}$$

c) Cuando encontremos la primera A , la cambiamos por una a y volvemos hacia atrás buscando γ . Para ello, reutilizaremos Z , añadiendo una nueva regla a Z :

$$\begin{aligned} AH' &\rightarrow aZ \\ Z\gamma &\rightarrow H\gamma \end{aligned}$$

H realizará este procedimiento de forma iterativa, hasta llegar a la última B , operación sensible, por lo que para realizarla usaremos una nueva variable \overline{H} :

$$\begin{aligned} \beta BH &\rightarrow \overline{H}\beta \quad (\text{Cojo la última}) \\ b\overline{H} &\rightarrow \overline{H}b \\ a\overline{H} &\rightarrow \overline{H}a \end{aligned}$$

Como se trata de la última B , sólo vamos a sustituir una A en caso de que sea la última, con lo que:

$$\lambda A\overline{H} \rightarrow a\psi$$

Donde ψ es la última variable, la cual se encarga de limpiar toda la palabra limpiando las variables.

Hemos tenido en cuenta el caso en el que $n = m^2$, pero faltan dos casos por considerar:

- $n < m^2$, es decir, hay más B s que A s. En dicho caso, la última A a sustituir no será consecuencia de quitar la última B , sino una anterior, con lo que quedará una B por eliminar (que podrá ser o no la última) que no tenga una A para sustituir. Como $n < m^2$, la palabra es válida. Añadimos por tanto las reglas:

$$\begin{aligned} \lambda H' &\rightarrow \psi \\ \lambda \overline{H} &\rightarrow \psi \end{aligned}$$

Y la variable ψ deberá ser la encargada de eliminar las B s sobrantes.

- $n > m^2$, es decir, hay más As que Bs . En dicho caso, cuando borremos la última B , no podremos sustituir su A asociada, ya que la regla para realizar esto es $\lambda A\bar{H} \rightarrow a\psi$, con lo que es necesario que sólo quede una A , cosa que no sucede. En dicho caso, nos quedaremos con una palabra de la forma:

$$\lambda A \dots A\bar{H}b \dots b\beta\gamma$$

que será imposible sustituir por un símbolo terminal (no tenemos ninguna regla que lo permita). De esta forma, la gramática impide generar palabras que no se encuentren en el lenguaje.

Finalmente, mostramos el funcionamiento de la variable ψ , cuya única funcionalidad es eliminar las variables β y γ una vez sustituidas todas las As por as (en caso de que la palabra sea válida), y eliminando también las posibles Bs sobrantes:

$$\begin{aligned} \psi a &\rightarrow a\psi \\ \psi b &\rightarrow b\psi \\ \psi \beta &\rightarrow \psi \\ \psi B &\rightarrow \psi \\ \psi \gamma &\rightarrow \varepsilon \quad (\text{Hemos terminado}) \end{aligned}$$

Para esta gramática no mostraremos un ejemplo de su funcionamiento. Si algún lector está dispuesto a realizar un ejemplo de generación de una palabra perteneciente al lenguaje o el intento de una palabra que no pertenezca al lenguaje (con la finalidad de ver que dicha palabra no podrá ser generada), será de nuestro agrado subir el ejemplo a este documento.

1.1.2. Preguntas Tipo Test

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

1. Si un lenguaje es generado por una gramática dependiente del contexto, entonces dicho lenguaje no es independiente del contexto.

Falso, los lenguajes generados por gramáticas independientes del contexto están contenidos en los generados por las gramáticas dependientes del contexto, por lo que muchas veces si tenemos un lenguaje generado por una gramática independiente del contexto, podremos encontrar una dependiente del contexto que nos genere el mismo lenguaje.

Por ejemplo, el lenguaje $L = \{a^i b \mid i \in \mathbb{N}\}$ puede generarse por una gramática dependiente del contexto como lo es: $G = (V, T, P, S)$ con

$$\begin{aligned} V &= \{S, B\} \\ T &= \{a, b\} \\ P &= \left\{ \begin{array}{ll} S & \rightarrow aBb \\ aBb & \rightarrow aaBb \mid \varepsilon \end{array} \right\} \end{aligned}$$

Pero sin embargo, podemos dar una gramática regular (y por tanto, independiente de contexto) para este lenguaje, como por ejemplo $G' = (V, T, P', S)$ con:

$$P' = \left\{ \begin{array}{l} S \rightarrow aS \mid B \\ B \rightarrow b \end{array} \right\}$$

2. Los alfabetos tienen siempre un número finito de elementos, pero los lenguajes, incluso si el alfabeto tiene sólo un símbolo, tienen infinitas palabras.

Falso, dado un alfabeto A , el conjunto relacionado con él y que siempre tiene infinitas palabras es el conjunto de todas las palabras de A , A^* ; salvo cuando A es vacío.

Sin embargo, podemos dar un ejemplo de alfabeto con un lenguaje finito:

$$A = \{a\} \quad L = \{aa\} \subseteq A^*$$

3. Si L es un lenguaje no vacío, entonces L^* es infinito.

Verdadero, ya que si L es no vacío, entonces existirá una palabra $u \in L$, luego $u^i \in L^* \forall i \in \mathbb{N}$. Podemos por tanto construir una aplicación inyectiva $f: \mathbb{N} \rightarrow L^*$ que asigna $n \in \mathbb{N}$ en u^n . Concluimos por tanto que L^* es infinito.

4. Todo lenguaje con un número finito de palabras es regular e independiente del contexto.

Verdadero, como todo lenguaje regular es a su vez independiente del contexto, será suficiente con probar que todo lenguaje finito es regular. Para ello, dado un lenguaje finito cualquiera $L = \{u_1, u_2, \dots, u_n\}$, podemos construir una gramática generativa de tipo 3 que nos genere dicho lenguaje (suponiendo que el lenguaje contiene palabras de un cierto alfabeto A):

$$G = (\{S\}, A, P, S)$$

$$P = \{ S \rightarrow u_1 \mid u_2 \mid \dots \mid u_n \}$$

Con lo que L será regular.

5. Si L es un lenguaje, entonces siempre L^* es distinto de L^+ .

Falso, si $\varepsilon \in L$, como por ejemplo ocurre con el lenguaje $L = \{\varepsilon\}$, entonces tendremos que $L^* = L^+$.

6. $L\emptyset = L$.

Falso (salvo si $L = \emptyset$, en cuyo caso es trivial):

$$L\emptyset = \{uv \mid u \in L, v \in \emptyset\}$$

En el caso de ser $L \neq \emptyset$, si $u \in L$ estaríamos diciendo que toda palabra de L puede descomponerse en dos, una de L y otra del vacío, lo que nos lleva a una contradicción. Concluimos que $L\emptyset = \emptyset$.

7. Si A es un alfabeto, la aplicación que transforma cada palabra $u \in A^*$ en su inversa es un homomorfismo de A^* en A^* .

Falso, como contraejemplo, trabajaremos con el alfabeto $A = \{a, b\}$ y supongamos que dicha aplicación es f , por ser homomorfismo, esta debe cumplir que:

$$f(uv) = f(u)f(v) \quad \forall u, v \in A^*$$

Sin embargo, si tomamos $u = ab$ y $v = ba$, tenemos:

$$f(uv) = f(abba) = abba \neq baab = f(ab)f(ba) = f(u)f(v)$$

8. Si $\varepsilon \in L$, entonces $L^+ = L^*$.

Verdadero:

$$L^* = \bigcup_{i \geq 0} L^i = L^0 \cup L \cup \left(\bigcup_{i \geq 2} L^i \right) \stackrel{(*)}{=} L \cup \left(\bigcup_{i \geq 2} L^i \right) = \bigcup_{i \geq 1} L^i = L^+$$

Donde en $(*)$ hemos aplicado que $L^0 = \{\varepsilon\} \subseteq L = L^1$.

9. La transformación que a cada palabra sobre $\{0, 1\}$ le añade 00 al principio y 11 al final es un homomorfismo.

Falso, como contraejemplo, trabajaremos con el alfabeto $A = \{0, 1\}$ y supongamos que dicha aplicación es f , por ser homomorfismo, esta debería cumplir que:

$$f(uv) = f(u)f(v) \quad \forall u, v \in A^*$$

Sin embargo, si tomamos $u = v = 1$:

$$f(uv) = f(11) = 001111 \neq 0011100111 = f(1)f(1) = f(u)f(v)$$

10. Se puede construir un programa que tenga como entrada un programa y unos datos y que siempre nos diga si el programa leído termina para esos datos.

Falso, este problema es conocido como el problema de la parada, y es conocido que no es posible construir dicho programa.

11. La cabecera del lenguaje L siempre incluye a L .

Falso, $CAB(L)$ es el conjunto de prefijos de palabras de L y puede suceder que los prefijos de palabras de L no sean palabras de L . Como contraejemplo, consideramos $L = \{ab\}$, luego $CAB(L) = \{\varepsilon, a, ab\} \neq L$.

Notemos que si $\varepsilon \notin L$, entonces $CAB(L) \not\subseteq L$, ya que $\varepsilon \in CAB(L)$ para cualquier lenguaje no vacío L .

12. Un lenguaje nunca puede ser igual a su inverso.

Falso, todos los lenguajes unitarios (que contienen solo una palabra) son iguales a su inverso. Como contraejemplo más elaborado, cualquier lenguaje L que cumpla que

$$L \subseteq \{u \in A^* \mid u^{-1} = u\}$$

sirve de contraejemplo.

13. La aplicación que transforma cada palabra u sobre el alfabeto $\{0, 1\}$ en u^3 es un homomorfismo.

Falso, como contraejemplo, suponemos que la aplicación f transforma cada palabra u en u^3 . De ser un homomorfismo, se debería cumplir que:

$$f(uv) = f(u)f(v) \quad \forall u, v \in A^*$$

Sin embargo, si tomamos $u = 0$ y $v = 1$:

$$f(uv) = f(01) = 010101 \neq 000111 = f(0)f(1) = f(u)f(v)$$

De forma general, podemos decir que cualquier aplicación que transforme cada palabra u en u^n para $n \geq 2$ (la identidad sí que es un homomorfismo) no es un homomorfismo, por un razonamiento análogo al anterior.

14. El lenguaje que contiene sólo la palabra vacía es el elemento neutro para la concatenación de lenguajes.

Verdadero, sea $L_e = \{\varepsilon\}$, recordamos que:

$$L_1 L_2 = \{uv \mid u \in L_1, v \in L_2\}$$

Por tanto:

$$L_e L = \{\varepsilon v \mid v \in L\} = L$$

$$L L_e = \{v \varepsilon \mid v \in L\} = L$$

Para cualquier lenguaje L .

15. Si L es un lenguaje, en algunas ocasiones se tiene que $L^* = L^+$.

Verdadero, como hemos visto anteriormente, es condición suficiente (se deduce trivialmente que es necesaria, ya que $\varepsilon \in L^*$ siempre) que $\varepsilon \in L$.

16. Hay lenguajes con un número infinito de palabras que no son regulares.

Verdadero, sabemos que los lenguajes con un número finito de palabras son regulares, luego cualquier lenguaje que no sea regular deberá tener un número infinito de palabras. Como sabemos la existencia de lenguajes no regulares, como por ejemplo $L = \{a^i b^j a^i b^j \mid i, j \in \mathbb{N}\}$, entonces estos han de tener un número no finito de palabras.

17. Si un lenguaje tiene un conjunto infinito de palabras sabemos que no es regular.

Falso, el lenguaje $L = \{a^i \mid i \in \mathbb{N} \setminus \{0\}\}$ tiene un conjunto infinito de palabras y es regular, ya que podemos dar una gramática generativa de tipo 3:
 $G = (\{S\}, \{a\}, \{S \rightarrow aS, S \rightarrow a\}, S)$ que genera dicho lenguaje.

18. Si L es un lenguaje finito, entonces su cabecera ($CAB(L)$) también será finita.

Verdadero, supongamos que tenemos un lenguaje finito L formado por $n \in \mathbb{N}$ palabras y recordamos que $CAB(L)$ es el conjunto formado por todos los prefijos de palabras de L . Dada una palabra $u \in L$ con $|u| = m$, esta estará formada por m letras del alfabeto A :

$$u = a_1 a_2 \dots a_m \quad a_i \in A \quad \forall i \in \{1, \dots, m\}$$

Por lo que aceptará $m + 1$ prefijos distintos:

$$\varepsilon \quad a_1 \quad a_1 a_2 \quad \dots \quad a_1 a_2 \dots a_{m-1} \quad a_1 a_2 \dots a_m$$

No obstante, hemos de tener en cuenta que un mismo prefijo puede ser prefijo de dos palabras distintas de L . En particular, como ε es prefijo de todas las palabras, tendremos que, sin contar ε , cada palabra de L aportará m prefijos distintos. Por tanto, el conjunto $CAB(L)$ tendrá como máximo:

$$|CAB(L)| \leq \left(\sum_{u \in L} |u| \right) + 1$$

donde el $+1$ se debe a la presencia de ε en $CAB(L)$. Sea ahora $w \in L$ la palabra de L con mayor longitud, tenemos que

$$|CAB(L)| \leq \left(\sum_{u \in L} |u| \right) + 1 \leq \left(\sum_{i=1}^n |w| \right) + 1 = n \cdot |w| + 1$$

19. El conjunto de palabras sobre un alfabeto dado con la operación de concatenación tiene una estructura de monoide.

Verdadero, recordamos que un par conjunto-operación tiene estructura de monoide si:

- Se trata de una operación interna al conjunto, lo cual es cierto, ya que la concatenación de dos palabras cualquiera es una palabra.
- La operación es asociativa, algo que también cumple la concatenación.
- Existe un elemento neutro del conjunto para dicha operación, lo cual es cierto, debido a la existencia de ε en cualquier conjunto de palabras dado por un alfabeto.

20. La transformación entre el conjunto de palabras del alfabeto $\{0, 1\}$ que duplica cada símbolo (la palabra 011 se transforma en 001111) es un homomorfismo.

Verdadero, sea el alfabeto $A = \{0, 1\}$ y $f : A^* \rightarrow A^*$ la aplicación enunciada, definida por:

$$f(u) = a_1 a_1 a_2 a_2 \dots a_n a_n \quad \forall u = a_1 a_2 \dots a_n \quad a_i \in A \quad \forall i \in \{1, \dots, n\}$$

Para ver que sea un homomorfismo, debemos comprobar que se cumpla

$$f(uv) = f(u)f(v) \quad \forall u, v \in A^*$$

Para ello, sean $u, v \in A^*$ palabras de longitud $n, m \in \mathbb{N}$ respectivamente, entonces tendremos que

$$\begin{aligned} u &= a_1 a_2 \dots a_n & a_i &\in A \quad \forall i \in \{1, \dots, n\} \\ v &= b_1 b_2 \dots b_m & b_i &\in A \quad \forall i \in \{1, \dots, m\} \end{aligned}$$

De esta forma:

$$\begin{aligned} f(uv) &= f(a_1 a_2 \dots a_n b_1 b_2 \dots b_m) = a_1 a_2 a_2 a_2 \dots a_n a_n b_1 b_1 b_2 b_2 \dots b_m b_m = \\ &= f(a_1 a_2 \dots a_n) f(b_1 b_2 \dots b_m) = f(u) f(v) \end{aligned}$$

Con lo que f es un homomorfismo.

21. Si f es un homomorfismo entre palabras del alfabeto A_1 en palabras del alfabeto de A_2 , entonces si conocemos $f(a)$ para cada $a \in A_1$ se puede calcular $f(u)$ para cada palabra $u \in A_1^*$.

Verdadero, sea $f : A_1^* \rightarrow A_2^*$ un homomorfismo, este cumplirá por tanto que:

$$f(uv) = f(u)f(v) \quad \forall u, v \in A_1^*$$

Puede demostrarse fácilmente por inducción que si tenemos una palabra $u \in A_1^*$ formada por n letras del alfabeto:

$$u = a_1 a_2 \dots a_n \quad a_i \in A \quad \forall i \in \{1, \dots, n\}$$

entonces, se tiene que

$$f(u) = f(a_1) f(a_2) \dots f(a_n)$$

Por tanto, el enunciado es cierto.

22. Si A es un alfabeto, la aplicación que transforma cada palabra $u \in A^*$ en su inversa es un homomorfismo de A^* en A^* .

Falso, la pregunta es idéntica a la pregunta 7.

23. Si $\varepsilon \in L$, entonces $L^+ = L^*$.

Verdadero, la pregunta es idéntica a la pregunta 8.

24. Si f es un homomorfismo, entonces necesariamente se verifica $f(\varepsilon) = \varepsilon$.

Verdadero, sea $f : A^* \rightarrow A^*$ un homomorfismo y consideramos $u \in A^*$. Por ser f un homomorfismo, tenemos que:

$$f(u) = f(\varepsilon u) = f(\varepsilon)f(u)$$

Con lo que $f(\varepsilon) = \varepsilon$.

25. Si A es un alfabeto, entonces A^+ no incluye nunca la palabra vacía.

Considerando A solamente como alfabeto Verdadero. Por definición del operador $^+$ aplicado a alfabetos, tenemos que:

$$A^+ = A^* \setminus \{\varepsilon\}$$

Considerando A como lenguaje Verdadero. Por definición del operador $^+$ aplicado a lenguajes, tenemos que:

$$A^+ = \bigcup_{i \geq 1} A^i$$

Por tanto, si $u \in A^+$, entonces $\exists n \in \mathbb{N}$ tal que $u \in A^n$, por lo que $u = a_1 a_2 \dots a_n$ con $a_i \in A \forall i \in \{1, \dots, n\}$. Como $a_i \in A$, entonces $a_i \neq \varepsilon \forall i \in \{1, \dots, n\}$, con lo que $u \neq \varepsilon$.

26. Es posible diseñar un algoritmo que lea un lenguaje cualquiera sobre el alfabeto $\{0, 1\}$ y nos diga si es regular o no.

Esto es falso, y se verá en la asignatura de Modelos Avanzados de Computación.

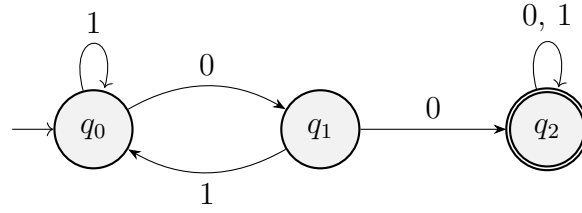


Figura 1.2: Autómata Finito Determinista del Ejercicio 1.2.1

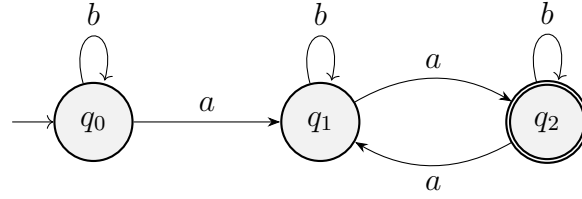


Figura 1.3: Autómata Finito Determinista del Ejercicio 1.2.2

1.2. Autómatas Finitos

Ejercicio 1.2.1. Considera el siguiente Autómata Finito Determinista (AFD) dado por $M = (Q, A, \delta, q_0, F)$, donde:

- $Q = \{q_0, q_1, q_2\}$
- $A = \{0, 1\}$
- La función de transición viene dada por:

$$\begin{array}{ll}
 \delta(q_0, 0) = q_1, & \delta(q_0, 1) = q_0 \\
 \delta(q_1, 0) = q_2, & \delta(q_1, 1) = q_0 \\
 \delta(q_2, 0) = q_2, & \delta(q_2, 1) = q_2
 \end{array}$$

- $F = \{q_2\}$

Describe informalmente el lenguaje aceptado.

Su representación gráfica está en la Figura 1.2.

Tenemos que el lenguaje aceptado por el autómata es el conjunto de todas las palabras que contienen la cadena 00 como subcadena. Es decir,

$$L = \{u_1 00 u_2 \in \{0, 1\}^* \mid u_1, u_2 \in \{0, 1\}^*\}.$$

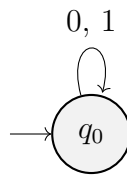
Ejercicio 1.2.2. Dado el AFD de la Figura 1.3, describir el lenguaje aceptado por dicho autómata.

El lenguaje aceptado por el autómata es el conjunto de todas las palabras que contienen un número par de a 's. Es decir,

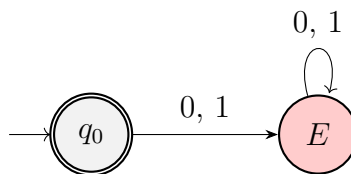
$$L = \{u \in \{a, b\}^* \mid n_a(u) \text{ es par, } n_a(u) > 0\},$$

Ejercicio 1.2.3. Dibujar AFDs que acepten los siguientes lenguajes con alfabeto $\{0, 1\}$:

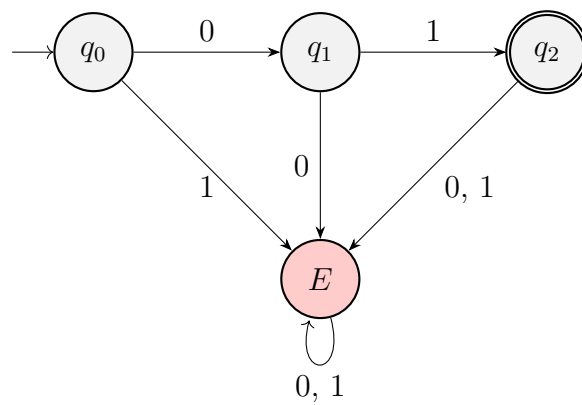
1. El lenguaje vacío,



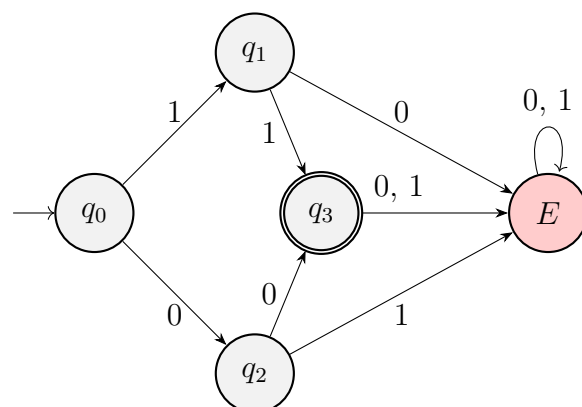
2. El lenguaje formado por la palabra vacía, es decir, $\{\varepsilon\}$,



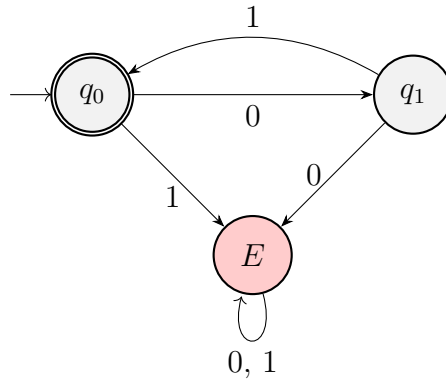
3. El lenguaje formado por la palabra 01, es decir, $\{01\}$,



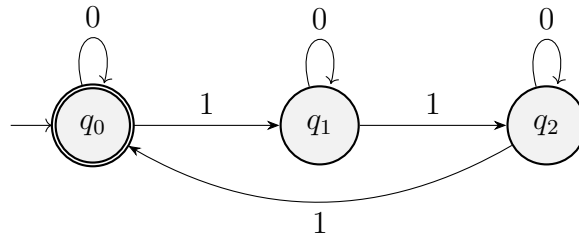
4. El lenguaje $\{11, 00\}$,



5. El lenguaje $\{(01)^i \mid i \geq 0\}$,



6. El lenguaje formado por las cadenas con 0's y 1's donde el número de unos es divisible por 3.



Ejercicio 1.2.4. Obtener a partir de la gramática regular $G = (\{S, B\}, \{1, 0\}, P, S)$, con

$$P = \begin{cases} S \rightarrow 110B \\ B \rightarrow 0B \mid 1B \mid \varepsilon \end{cases}$$

un AFND que reconozca el lenguaje generado por esa gramática.

El autómata obtenido es el de la Figura 1.4.

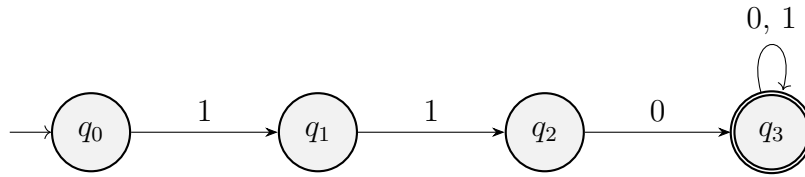


Figura 1.4: Autómata Finito No Determinista del Ejercicio 1.2.4

Ejercicio 1.2.5. Dada la gramática regular $G = (\{S\}, \{1, 0\}, P, S)$, con

$$P = \{S \rightarrow S10, S \rightarrow 0\},$$

obtener un AFD que reconozca el lenguaje generado por esa gramática.

El lenguaje es:

$$L = \{0(10)^n \mid n \in \mathbb{N} \cup \{0\}\}.$$

El autómata obtenido es el de la Figura 1.5.

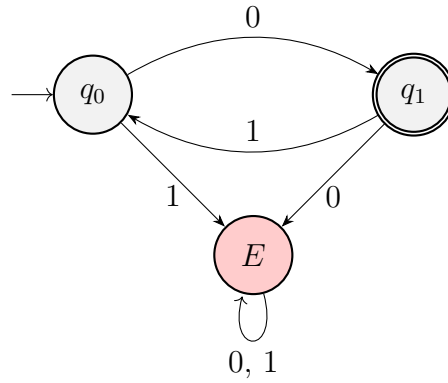


Figura 1.5: Autómata Finito Determinista del Ejercicio 1.2.5

Ejercicio 1.2.6. Construir un AFND o AFD (dependiendo del caso) que acepte las cadenas $u \in \{0, 1\}^*$ que:

1. AFND. Contengan la subcadena 010.

El autómata obtenido es el de la Figura 1.6.

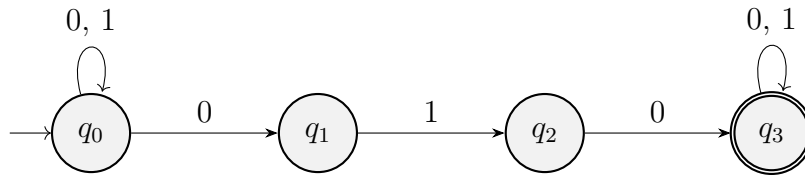


Figura 1.6: Autómata Finito No Determinista del Ejercicio 1.2.6 apartado 1.

2. AFND. Contengan la subcadena 110.

El autómata obtenido es el de la Figura 1.7.

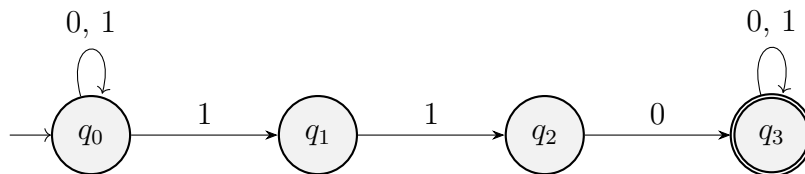


Figura 1.7: Autómata Finito No Determinista del Ejercicio 1.2.6 apartado 2.

3. AFD. Contengan simultáneamente las subcadenas 010 y 110.

El estado q_0 representa que no se ha empezado ninguna de las subcadenas, y el estado q_F representa que se han encontrado ambas cadenas. Hay dos opciones:

Opción 1 Primero se lee 010 y luego 110. Son los siguientes estados:

- q_0 : Estado inicial, no ha empezado la subcadena 010.
- q_1 : Se ha leído el 0 de la subcadena 010.
- q_2 : Se ha leído la subcadena 01 de la subcadena 010.
- q_3 : Se ha leído la subcadena 010. No ha empezado la subcadena 110.

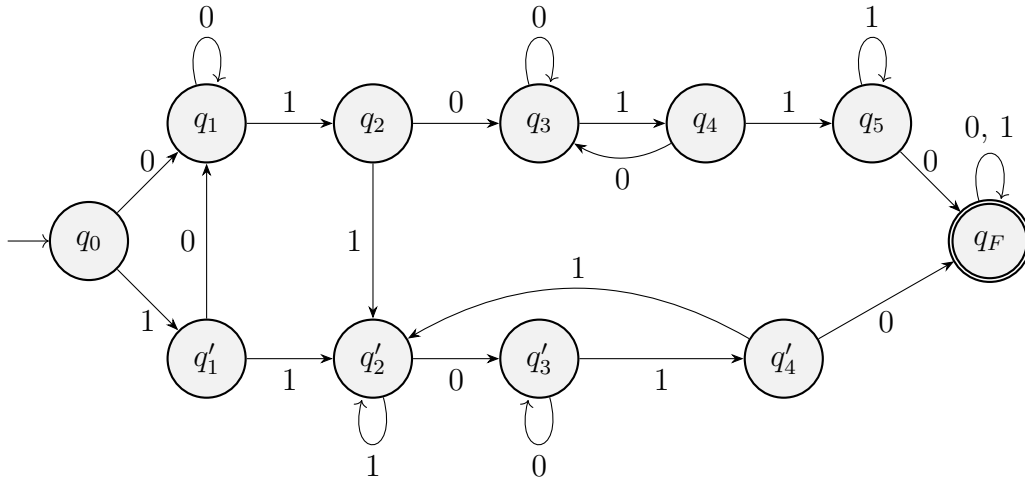


Figura 1.8: Autómata Finito Determinista del Ejercicio 1.2.6 apartado 3.

- q_4 : Se ha leído el 1 de la subcadena 110.
- q_5 : Se ha leído la subcadena 11 de la subcadena 110.
- q_F : Se ha leído la subcadena 110. Se han leído ambas subcadenas.

Opción 2 Primero se lee 110 y luego 010. Son los siguientes estados:

- q_0 : Estado inicial, no ha empezado la subcadena 110.
- q'_1 : Se ha leído el 1 de la subcadena 110.
- q'_2 : Se ha leído la subcadena 11 de la subcadena 110.
- q'_3 : Se ha leído la subcadena 110. Se ha leído el 0 de la subcadena 010. Notemos que en este caso podemos agruparlo, puesto que el último carácter de la subcadena 110 es el mismo que el primero de la subcadena 010.
- q'_4 : Se ha leído la subcadena 01 de la subcadena 010.
- q_F : Se ha leído la subcadena 010. Se han leído ambas subcadenas.

El autómata obtenido es el de la Figura 1.8.

Ejercicio 1.2.7. Construir un AFD que acepte el lenguaje generado por la siguiente gramática:

$$S \rightarrow AB, \quad A \rightarrow aA, \quad A \rightarrow c, \quad B \rightarrow bBb, \quad B \rightarrow b.$$

El lenguaje generado por la gramática es:

$$L = \{a^n cb^{2m+1} \mid n, m \in \mathbb{N} \cup \{0\}\}.$$

El autómata obtenido es el de la Figura 1.9.

Ejercicio 1.2.8. Construir un AFD que acepte el lenguaje $L \subseteq \{a, b, c\}^*$ de todas las palabras con un número impar de ocurrencias de la subcadena abc .

El autómata tiene los siguientes estados:

- q_0 : Llevo un número par de ocurrencias de abc , y no he empezado la siguiente.

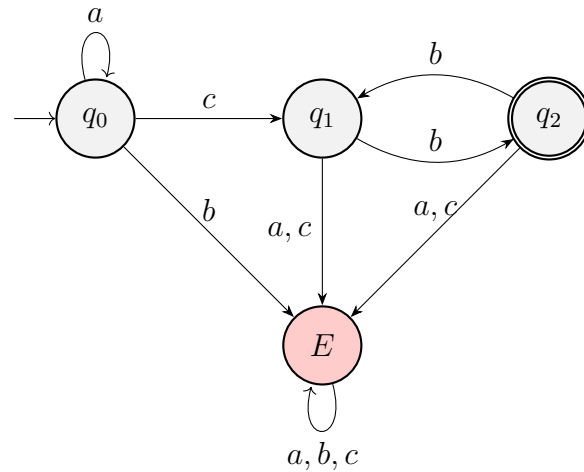


Figura 1.9: Autómata Finito Determinista del Ejercicio 1.2.7

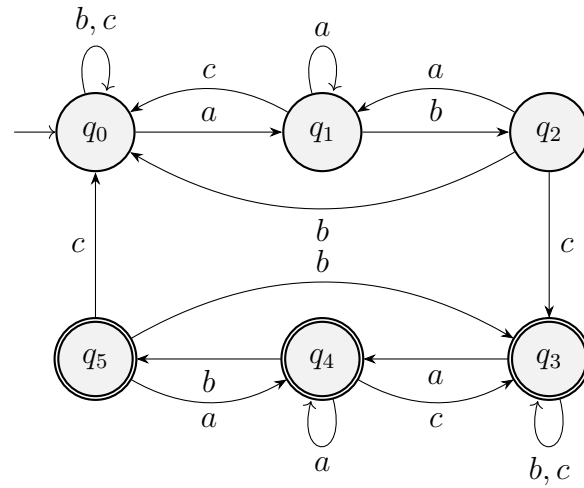


Figura 1.10: Autómata Finito Determinista del Ejercicio 1.2.8

- q_1 : Acabo de empezar una ocurrencia impar de abc , llevo solo una a .
- q_2 : Estoy en una ocurrencia impar de abc , llevo ab .
- q_3 : Llevo un número impar de ocurrencias de abc , y no he empezado la siguiente.
- q_4 : Acabo de empezar una ocurrencia par de abc , llevo solo una a .
- q_5 : Estoy en una ocurrencia par de abc , llevo ab .

El autómata obtenido es el de la Figura 1.10.

Ejercicio 1.2.9. Sea L el lenguaje de todas las palabras sobre el alfabeto $\{0, 1\}$ que no contienen dos 1s que estén separados por un número impar de símbolos. Describir un AFD que acepte este lenguaje.

Sea $u \in L$. Veamos que, a lo sumo, puede tener dos 1's. Supongamos por reducción al absurdo que tiene tres 1's. Entonces, entre la primera y la segunda hay un

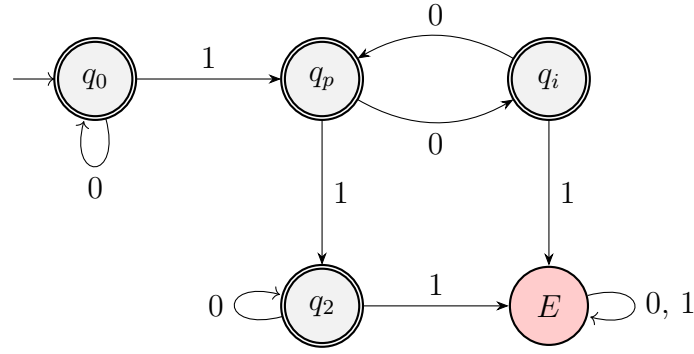


Figura 1.11: Autómata Finito Determinista del Ejercicio 1.2.9.

número impar de símbolos, y entre la segunda y la tercera hay un número impar de símbolos. Por lo tanto, entre el primer y el tercer 1 hay:

- Un número par de símbolos antes del segundo 1.
- El segundo 1.
- Un número par de símbolos entre el segundo y el tercer 1.

Por tanto, como el número de símbolos entre el primer y el tercer 1 es impar, entonces $u \notin L$. Por lo tanto, u tiene a lo sumo dos 1's.

Por tanto, los estados son:

- q_0 : No se ha introducido ningún 1.
- q_p : Se ha introducido un 1, después de él y antes del siguiente 1 hay un número par de símbolos.
- q_i : Se ha introducido un 1, después de él y antes del siguiente 1 hay un número impar de símbolos.
- q_2 : Se han introducido dos 1's, y no se ha introducido ningún otro.
- E : Estado de error.

El autómata obtenido es el de la Figura 1.11.

Ejercicio 1.2.10. Dada la expresión regular $(a+\varepsilon)b^*$, encontrar un AFND asociado y, a partir de este, calcular un AFD que acepte el lenguaje.

El AFND con transiciones nulas obtenido (siguiendo el algoritmo) es el de la Figura 1.12.

Podemos simplificar este autómata para que así la transición al AFD sea más sencilla. El autómata simplificado es el de la Figura 1.13.

A partir de este autómata simplificado, obtenemos el AFD de la Figura 1.14.

Ejercicio 1.2.11. Obtener una expresión regular para el lenguaje complementario al aceptado por la gramática

$$S \rightarrow abA \mid B \mid baB \mid \varepsilon, \quad A \rightarrow bS \mid b, \quad B \rightarrow aS.$$

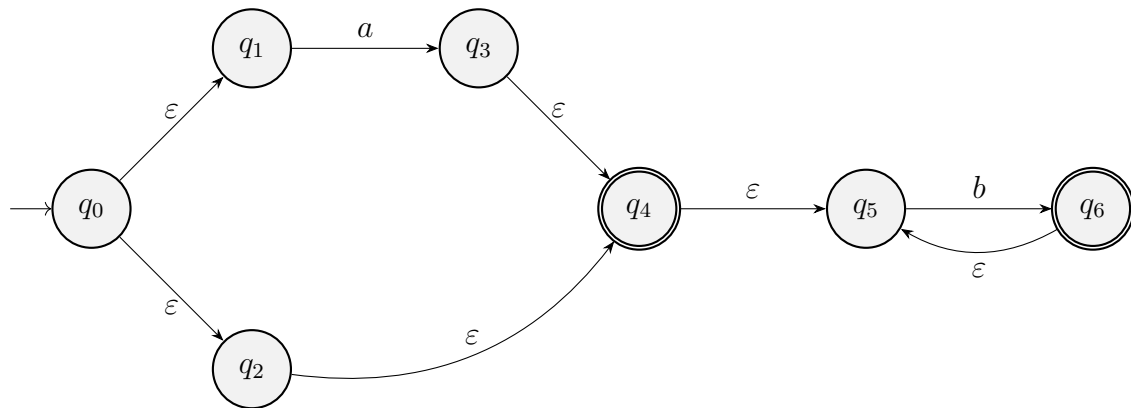


Figura 1.12: Autómata Finito No Determinista algorítmico del Ejercicio 1.2.10.

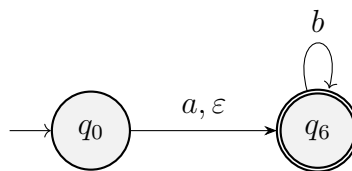


Figura 1.13: Autómata Finito No Determinista simplificado del Ejercicio 1.2.10.

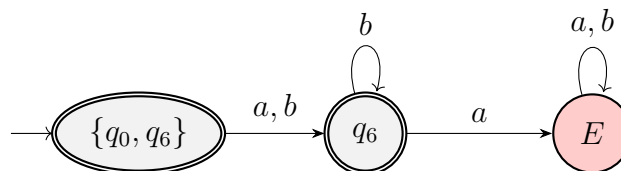


Figura 1.14: Autómata Finito Determinista del Ejercicio 1.2.10.

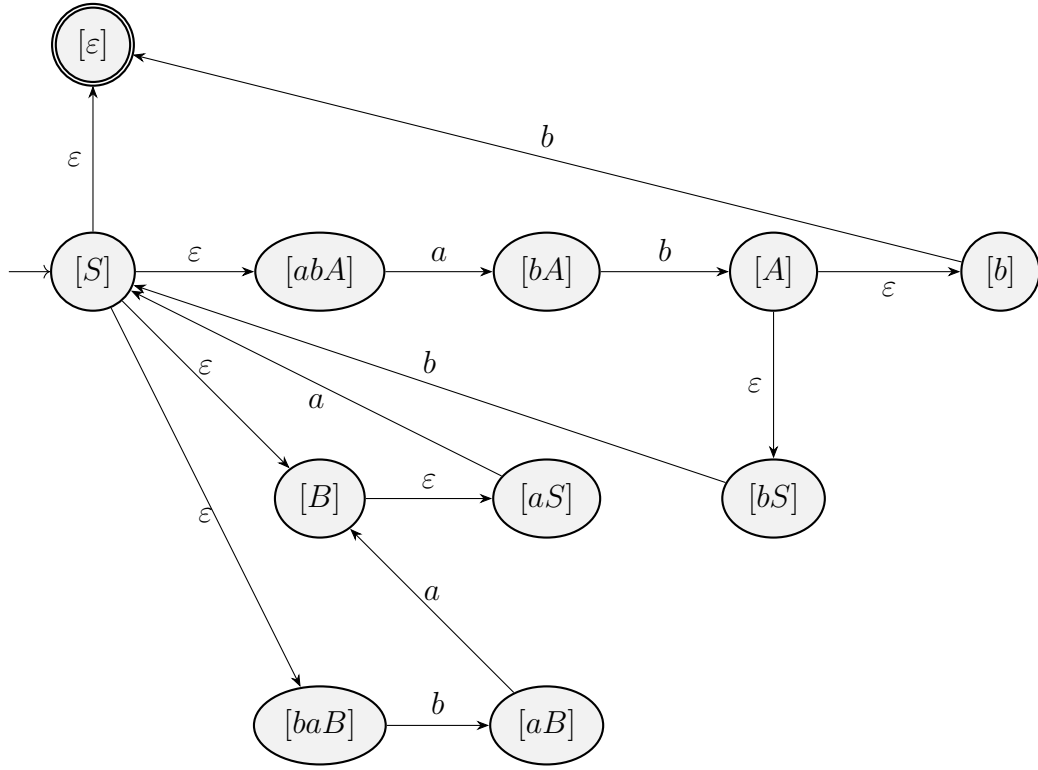


Figura 1.15: Autómata Finito Determinista del lenguaje $\mathcal{L}(S)$ del Ejercicio 1.2.11.

Observación. Construir un AFD asociado.

Esta gramática es lineal por la derecha. Algorítmicamente, obtenemos el autómata de la Figura 1.15 para el lenguaje generado por S , $\mathcal{L}(S)$.

Ahora, tendríamos que eliminar las transiciones nulas para poder así aplicar el algoritmo para hallar la expresión regular. Esto no es sencillo, por lo que vamos a intentar obtener de forma directa el AFD. Para ello, la gramática dada genera el mismo lenguaje que las siguientes reglas de producción, donde hemos eliminado la variable B :

$$S \rightarrow abA \mid aS \mid baaS \mid \varepsilon, \quad A \rightarrow bS \mid b$$

Eliminamos ahora la variable A :

$$S \rightarrow abbS \mid abb \mid aS \mid baaS \mid \varepsilon$$

Veamos ahora que la regla $S \rightarrow abb$ no es relevante, ya que podemos obtenerla a partir de $S \rightarrow abbS$ y $S \rightarrow \varepsilon$. Por tanto, la gramática dada inicialmente genera el mismo lenguaje que si estas fuesen las reglas de producción:

$$S \rightarrow abbS \mid aS \mid baaS \mid \varepsilon$$

Por tanto, vemos que:

$$\mathcal{L}(G) = \{abb, a, baa\}^*.$$

En consecuencia, la expresión regular asociada a $\mathcal{L}(G)$ es:

$$(abb + a + baa)^*$$

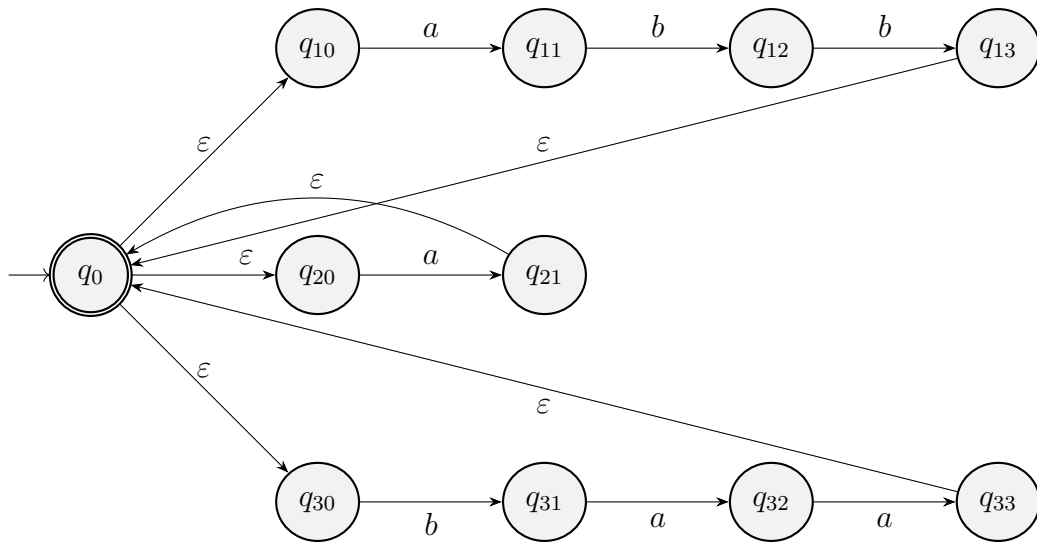


Figura 1.16: AFND asociado a $\mathcal{L}(G)$ del Ejercicio 1.2.11.

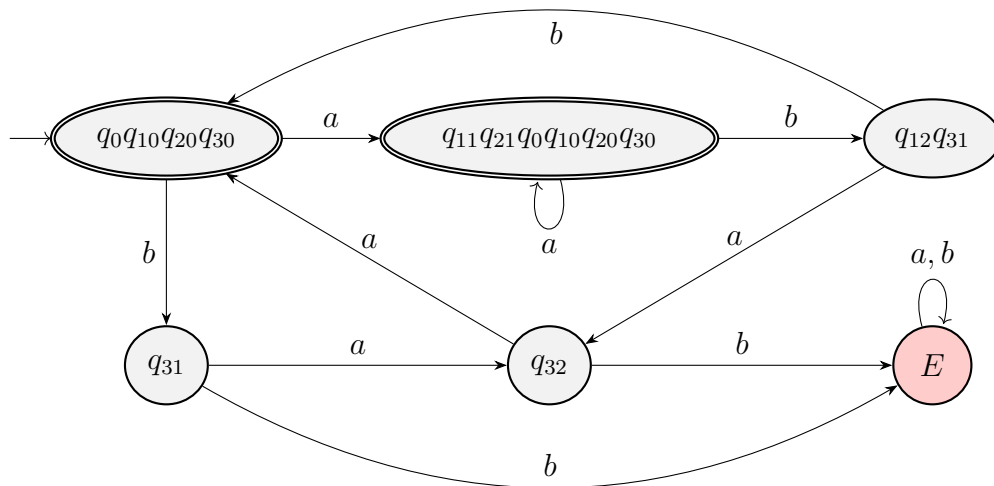


Figura 1.17: AFD asociado a $\mathcal{L}(G)$ del Ejercicio 1.2.11.

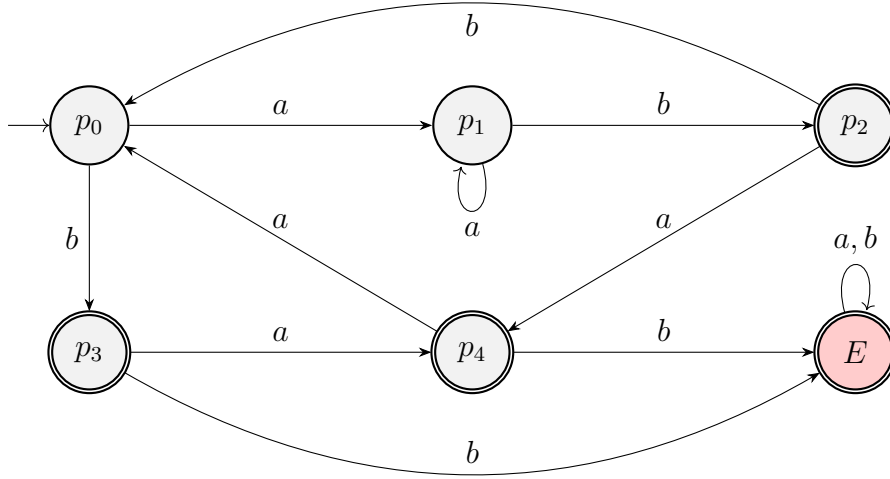


Figura 1.18: Autómata Finito Determinista del lenguaje $\overline{\mathcal{L}(G)}$ del Ejercicio 1.2.11.

El AFND asociado a esta expresión regular es el de la Figura 1.16.

Convirtiendo este autómata en un AFD, obtenemos el de la Figura 1.17.

Para que buscar la expresión regular del lenguaje complementario de $\mathcal{L}(G)$ sea más cómoda, cambiaremos la notación de cada estado. El AFD del lenguaje complementario de $\mathcal{L}(G)$ es el de la Figura 1.18.

Buscamos ahora una expresión para $\overline{\mathcal{L}(G)}$. Resolvemos el siguiente sistema:

$$\begin{cases} p_0 &= ap_1 + bp_3 \\ p_1 &= ap_1 + bp_2 \\ p_2 &= ap_4 + bp_0 + \varepsilon \\ p_3 &= ap_4 + bE + \varepsilon \\ p_4 &= ap_0 + bE + \varepsilon \\ E &= aE + bE + \varepsilon \end{cases}$$

De la última ecuación, obtenemos que $E = (a + b)^*$. El sistema queda:

$$\begin{cases} p_0 &= ap_1 + bp_3 \\ p_1 &= ap_1 + bp_2 \\ p_2 &= ap_4 + bp_0 + \varepsilon \\ p_3 &= ap_4 + b(a + b)^* + \varepsilon \\ p_4 &= ap_0 + b(a + b)^* + \varepsilon \end{cases}$$

Sustituyendo p_4 , obtenemos:

$$\begin{cases} p_0 &= ap_1 + bp_3 \\ p_1 &= ap_1 + bp_2 \\ p_2 &= a[ap_0 + b(a + b)^* + \varepsilon] + bp_0 + \varepsilon = (aa + b)p_0 + ab(a + b)^* + a + \varepsilon \\ p_3 &= a[ap_0 + b(a + b)^* + \varepsilon] + b(a + b)^* + \varepsilon = aap_0 + (a + \varepsilon)(b(a + b)^* + \varepsilon) \end{cases}$$

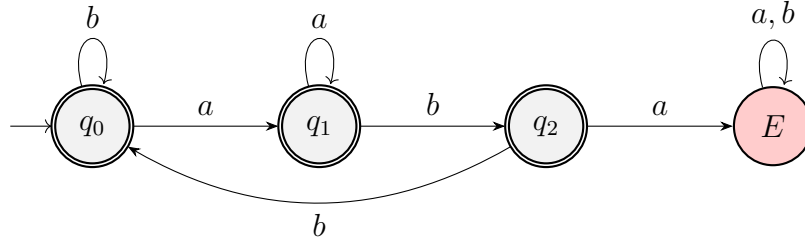


Figura 1.19: AFD del lenguaje del Ejercicio 1.2.12 apartado 3.

Tenemos que:

$$p_1 = ap_1 + bp_2 = a^*bp_2 = a^*b[(aa + b)p_0 + ab(a + b)^* + a + \varepsilon]$$

Sustituyendo, tenemos que:

$$\begin{aligned}
 p_0 &= a[a^*b[(aa + b)p_0 + ab(a + b)^* + a + \varepsilon]] + b[aap_0 + (a + \varepsilon)(b(a + b)^* + \varepsilon)] = \\
 &= a^+b(aa + b)p_0 + a^+bab(a + b)^* + a^+ba + a^+b + baap_0 + b(a + \varepsilon)(b(a + b)^* + \varepsilon) = \\
 &= [a^+b(aa + b) + baa]p_0 + a^+bab(a + b)^* + a^+ba + a^+b + b(a + \varepsilon)(b(a + b)^* + \varepsilon) \stackrel{(*)}{=} \\
 &\stackrel{(*)}{=} [a^+b(aa + b) + baa]^*[a^+bab(a + b)^* + a^+ba + a^+b + b(a + \varepsilon)(b(a + b)^* + \varepsilon)]
 \end{aligned}$$

donde en $(*)$ hemos aplicado el Lema de Arden. Por tanto, la expresión regular asociada a $\mathcal{L}(G)$ es:

$$[a^+b(aa + b) + baa]^*[a^+bab(a + b)^* + a^+ba + a^+b + b(a + \varepsilon)(b(a + b)^* + \varepsilon)]$$

Ejercicio 1.2.12. Dar expresiones regulares para los lenguajes sobre el alfabeto $\{a, b\}$ dados por las siguientes condiciones:

1. Palabras que no contienen la subcadena a ,

$$b^*$$

2. Palabras que no contienen la subcadena ab .

$$b^*a^*$$

3. Palabras que no contienen la subcadena aba .

Este lenguaje viene descrito por el autómata de la Figura 1.19.

Obtenemos la expresión regular asociada al lenguaje del autómata de la Figura 1.19.

$$\begin{cases}
 q_0 &= bq_0 + aq_1 + \varepsilon \\
 q_1 &= aq_1 + bq_2 + \varepsilon \\
 q_2 &= bq_0 + aE + \varepsilon \\
 E &= aE + bE = (a + b)E + \emptyset
 \end{cases}$$

Usando el Lema de Arden, obtenemos que $E = \emptyset(a + b)^* = \emptyset$. Sustituyendo, obtenemos:

$$\begin{cases} q_0 &= bq_0 + aq_1 + \varepsilon \\ q_1 &= aq_1 + bq_2 + \varepsilon \\ q_2 &= bq_0 + a\emptyset + \varepsilon = bq_0 + \varepsilon \end{cases}$$

Sustituyendo q_2 , obtenemos:

$$\begin{cases} q_0 &= bq_0 + aq_1 + \varepsilon \\ q_1 &= aq_1 + b(bq_0 + \varepsilon) + \varepsilon \end{cases}$$

Usando el Lema de Arden, obtenemos que:

$$q_1 = a^*[b(bq_0 + \varepsilon) + \varepsilon]$$

Sustituyendo en la primera ecuación, tenemos que:

$$\begin{aligned} q_0 &= bq_0 + aa^*[b(bq_0 + \varepsilon) + \varepsilon] + \varepsilon = \\ &= bq_0 + a^+[bbq_0 + b + \varepsilon] + \varepsilon = \\ &= (b + a^+bb)q_0 + a^+[b + \varepsilon] + \varepsilon \stackrel{(*)}{=} \\ &\stackrel{(*)}{=} (b + a^+bb)^*[a^+(b + \varepsilon) + \varepsilon] \end{aligned}$$

donde en (*) hemos aplicado el Lema de Arden. Por tanto, la expresión regular asociada al lenguaje del autómata de la Figura 1.19 es:

$$(b + a^+bb)^*[a^+(b + \varepsilon) + \varepsilon]$$

Ejercicio 1.2.13. Determinar si el lenguaje generado por la siguiente gramática es regular:

$$S \rightarrow AabB, \quad A \rightarrow aA \mid bA \mid \varepsilon, \quad B \rightarrow Bab \mid Bb \mid ab \mid b.$$

En caso de que lo sea, encontrar una expresión regular asociada.

Es directo ver que el lenguaje generado por la gramática tiene como expresión regular asociada:

$$(a + b)^*ab(ab + b)^+.$$

Por tanto, el lenguaje es regular.

Ejercicio 1.2.14. Sobre el alfabeto $A = \{0, 1\}$ realizar las siguientes tareas:

1. Describir un autómata finito determinista que acepte todas las palabras que contengan a 011 o a 010 (o las dos) como subcadenas.

Tenemos los siguientes estados:

- q_0 : No se ha empezado ninguna subcadena.

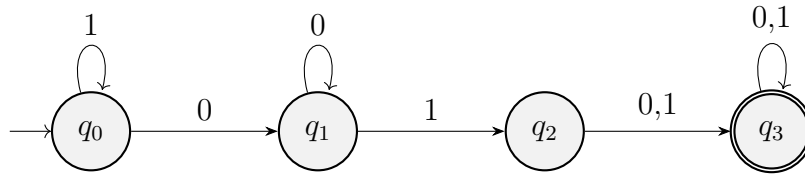


Figura 1.20: Autómata Finito Determinista del Ejercicio 1.2.14 apartado 1.

- $\underline{q_1}$: Se ha empezado una subcadena deseada. Tengo el carácter 0.
- $\underline{q_2}$: Se continúa la subcadena deseada. Tengo los caracteres 01.
- $\underline{q_3}$: Se ha encontrado la subcadena deseada. Tengo los caracteres 011 o 010.

El autómata obtenido es el de la Figura 1.20.

2. Describir un autómata finito determinista que acepte todas las palabras que empiecen o terminen (o ambas cosas) por 01.

Tenemos los siguientes estados:

- $\underline{q_0}$: No hemos leído nada.
- $\underline{q_1}$: Hemos empezado con un 0, por lo que puede comenzar por 01 (o terminar por 01).
- $\underline{q_2}$: Hemos empezado con 01, por lo que ya no hay más restricciones.
- $\underline{q_3}$: No hemos empezado por 01, por lo que ha de terminar por 01.
- $\underline{q_4}$: Ha de terminar por 01, y estamos en 0, por lo que si introduce un 1 puede terminar.
- $\underline{q_5}$: Ha de terminar por 01, y acabamos de leer 01, por lo que podemos terminar.

El autómata obtenido es el de la Figura 1.21.

3. Dar una expresión regular para el conjunto de las palabras en las que hay dos ceros separados por un número de símbolos que es múltiplo de 4 (los símbolos que separan los ceros pueden ser ceros y puede haber otros símbolos delante o detrás de estos dos ceros).

$$(0 + 1)^* \textcolor{red}{0} ((0 + 1)(0 + 1)(0 + 1)(0 + 1))^* \textcolor{red}{0} (0 + 1)^*$$

Notemos que los dos 0's en cuestión están marcados en rojo para facilitar la comprensión.

4. Dar una expresión regular para las palabras en las que el número de ceros es divisible por 4.

En un primer momento, podríamos pensar en:

$$(1^*01^*01^*01^*01^*)^*$$

No obstante, una palabra con 1's y sin 0's, que es aceptada por el lenguaje, no está contemplada en la expresión regular. La expresión regular correcta es:

$$(1^*01^*01^*01^*0)^*1^*$$

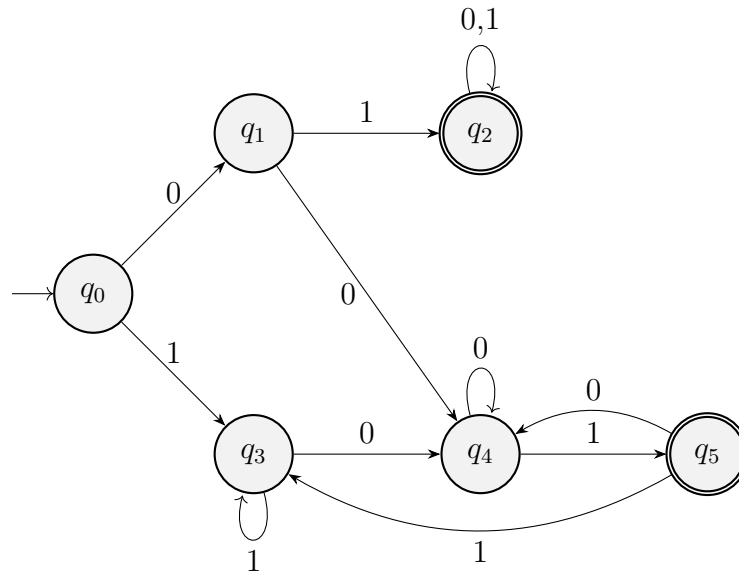


Figura 1.21: Autómata Finito Determinista del Ejercicio 1.2.14 apartado 2.

Ejercicio 1.2.15. Construye una gramática regular que genere el siguiente lenguaje:

$$L_1 = \{u \in \{0, 1\}^* \mid \text{el número de 1's y de 0's es impar}\}.$$

Tenemos los siguientes estados:

- $\underline{E_{01}}$: Tenemos un error en 0 y 1, ya que el número de 0's y de 1's es par.
- $\underline{E_0}$: Tenemos un error en 0, ya que el número de 0's es par. El número de 1's es impar.
- $\underline{E_1}$: Tenemos un error en 1, ya que el número de 1's es par. El número de 0's es impar.
- \underline{X} : No tenemos errores. El número de 0's y de 1's es impar.

La gramática obtenida es $G = (\{E_{01}, E_0, E_1, X\}, \{0, 1\}, P, E_{01})$, donde P es:

$$\begin{aligned} E_{01} &\rightarrow 0E_1 \mid 1E_0, \\ E_0 &\rightarrow 0X \mid 1E_{01}, \\ E_1 &\rightarrow 0E_{01} \mid 1X, \\ X &\rightarrow 0E_0 \mid 1E_1 \mid \varepsilon \end{aligned}$$

Ejercicio 1.2.16. Encuentra una expresión regular que represente el siguiente lenguaje:

$$L_2 = \{0^n 1^m \mid n \geq 1, m \geq 0, n \text{ múltiplo de } 3 \text{ y } m \text{ es par}\}.$$

La expresión regular es:

$$(000)^+(11)^*$$

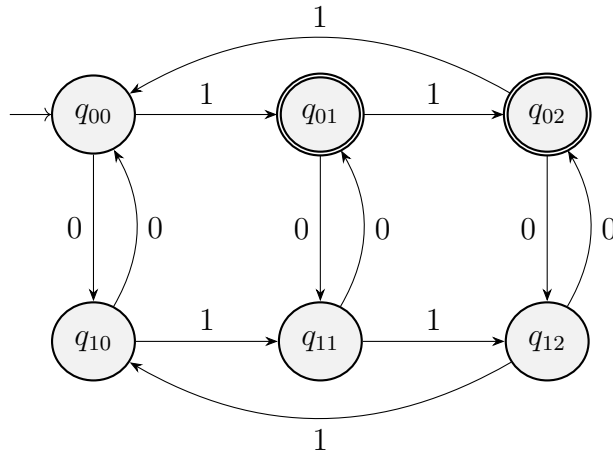


Figura 1.22: Autómata Finito Determinista del Ejercicio 1.2.17

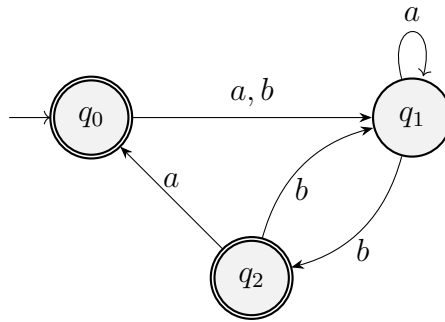


Figura 1.23: Autómata Finito Determinista del Ejercicio 1.2.18

Ejercicio 1.2.17. Diseña un autómata finito determinista que reconozca el siguiente lenguaje:

$$L_3 = \{u \in \{0, 1\}^* \mid \text{el número de 1's no es múltiplo de 3 y el número de 0's es par}\}.$$

Sean n_0 el número de 0's y n_1 el número de 1's.

Tenemos la siguiente disposición de estados:

- Los estados de arriba representan $n_0 \bmod 2 = 0$.
- Los estados de abajo representan $n_0 \bmod 2 = 1$.
- Los estados de la primera columna representan $n_1 \bmod 3 = 0$.
- Los estados de la segunda columna representan $n_1 \bmod 3 = 1$.
- Los estados de la tercera columna representan $n_1 \bmod 3 = 2$.

El estado q_{ij} representa $n_0 \bmod 2 = i$ y $n_1 \bmod 3 = j$.

El autómata obtenido es el de la Figura 1.22.

Ejercicio 1.2.18. Dar una expresión regular para el lenguaje aceptado por el autómata de la Figura 1.23.

Establecemos una ecuación por cada uno de los estados. El sistema inicial es:

$$\begin{cases} q_0 = \varepsilon + aq_1 + bq_1, \\ q_1 = aq_1 + bq_2, \\ q_2 = \varepsilon + aq_0 + bq_1. \end{cases}$$

Buscamos obtener la expresión regular asociada a q_1 :

$$\begin{aligned} q_1 &= aq_1 + b + baq_0 + bbq_1 = \\ &= baq_0 + b + (a + bb)q_1 \stackrel{(*)}{=} \\ &\stackrel{(*)}{=} (a + bb)^*(baq_0 + b) \end{aligned}$$

donde en $(*)$ hemos aplicado el Lema de Arden. Sustituyendo en la ecuación de q_0 obtenemos:

$$\begin{aligned} q_0 &= \varepsilon + (a + b)q_1 = \\ &= \varepsilon + (a + b)(a + bb)^*(baq_0 + b) = \\ &= \varepsilon + (a + b)(a + bb)^*b + (a + b)(a + bb)^*baq_0 \stackrel{(*)}{=} \\ &\stackrel{(*)}{=} ((a + b)(a + bb)^*ba)^*(\varepsilon + (a + b)(a + bb)^*b) \end{aligned}$$

donde, de nuevo, en $(*)$ hemos aplicado el Lema de Arden. Por tanto, la expresión regular asociada al autómata es:

$$((a + b)(a + bb)^*ba)^*(\varepsilon + (a + b)(a + bb)^*b).$$

Ejercicio 1.2.19. Dado el lenguaje

$$L = \{u110 \mid u \in \{1, 0\}^*\},$$

encontrar la expresión regular, la gramática lineal por la derecha, la gramática lineal por la izquierda y el AFD asociado.

La expresión regular es:

$$(0 + 1)^*110.$$

La gramática lineal por la derecha es $G = (\{S, A\}, \{0, 1\}, P, S)$, donde P es:

$$\begin{aligned} S &\rightarrow 0S \mid 1S \mid A \\ A &\rightarrow 110. \end{aligned}$$

La gramática lineal por la izquierda es $G = (\{S, A\}, \{0, 1\}, P', S)$, donde P' es:

$$\begin{aligned} S &\rightarrow X110 \\ X &\rightarrow X0 \mid X1 \mid \varepsilon. \end{aligned}$$

El AFD asociado es el de la Figura 1.24. Sus estados son:

- $\underline{q_0}$: No estoy en la cadena 110 final.

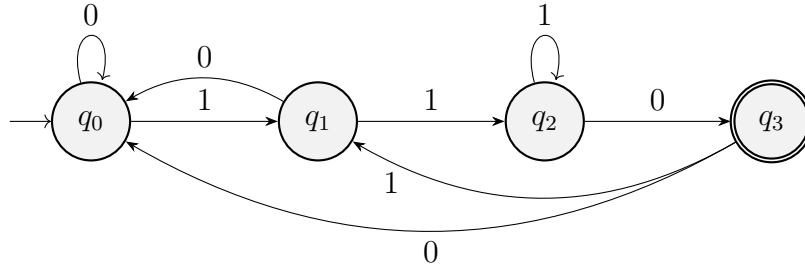


Figura 1.24: Autómata Finito Determinista del Ejercicio 1.2.19

- q₁: He leído un 1 de la cadena final.
- q₂: He leído un 11 de la cadena final.
- q₃: He leído un 110 de la cadena final.

Ejercicio 1.2.20. Dado un AFD, determinar el proceso que habría que seguir para construir una gramática lineal por la izquierda capaz de generar el Lenguaje aceptado por dicho autómata.

Sea $M = (Q, A, \delta, q_0, F)$ un AFD. Como Q es finito, podemos enumerar los estados como $Q = \{q_1, q_2, \dots, q_n\}$. La Gramática Lineal por la Izquierda asociada es $G = (Q \cup \{S\}, A, P, S)$, donde hemos supuesto $S \notin Q$ debido a nuestra enumeración de los estados. Las reglas de producción son:

$$P = \begin{cases} S \rightarrow q_i & \forall q_i \in F \\ q_i \rightarrow q_j a & \forall q_i, q_j \in Q, a \in A \mid \delta(q_j, a) = q_i \\ q_0 \rightarrow \varepsilon. \end{cases}$$

Notemos que lo que hacemos es invertir el autómata, obtener la gramática lineal por la derecha, y después invertir las reglas de producción de esta última.

Ejercicio 1.2.21. Construir un autómata finito determinista que acepte el lenguaje de todas las palabras sobre el alfabeto $\{0, 1\}$ que no contengan la subcadena 001. Construir una gramática regular por la izquierda a partir de dicho autómata.

Los estados son los siguientes:

- q₀: No se ha empezado la subcadena 001
- q₁: Se ha leído un 0 de la subcadena 001.
- q₂: Se ha leído un 00 de la subcadena 001.
- E: Se ha leído la subcadena 001, por lo que es el estado de error.

El autómata obtenido es el de la Figura 1.25.

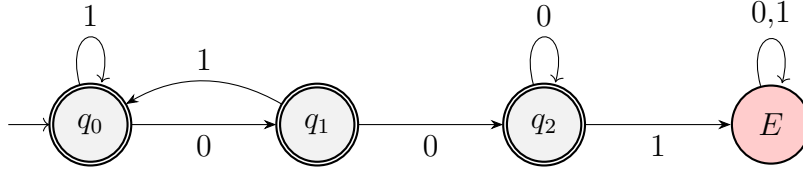


Figura 1.25: Autómata Finito Determinista del Ejercicio 1.2.21

Respecto a la gramática regular por la izquierda, usando el algoritmo descrito en el apartado anterior, tenemos que la gramática es $G = (Q \cup \{S\}, \{0, 1\}, P, S)$, donde P es:

$$P = \begin{cases} S & \rightarrow q_0 \mid q_1 \mid q_2, \\ q_0 & \rightarrow q_0 1 \mid q_1 1 \mid \epsilon, \\ q_1 & \rightarrow q_0 0, \\ q_2 & \rightarrow q_1 0 \mid q_2 0, \\ E & \rightarrow E 0 \mid E 1 \mid q_2 1, \end{cases}$$

Ejercicio 1.2.22. Sea $B_n = \{a^k \mid k \text{ es múltiplo de } n\}$. Demostrar que B_n es regular para todo n .

Fijado $n \in \mathbb{N}$, la expresión regular correspondiente es:

$$(a \overbrace{\dots}^{n \text{ veces}} a)^* = (a^n)^*$$

Equivalentemente, usando la notación de las expresiones regulares de UNIX, la expresión regular sería:

$$(a\{n\})^*$$

Ejercicio 1.2.23. Sea A un alfabeto. Decimos que $u \in A^*$ es un prefijo de $v \in A^*$ si existe $w \in A^*$ tal que $uw = v$. Decimos que u es un prefijo propio de v si además $u \neq v$ y $u \neq \epsilon$. Demostrar que si L es regular, también lo son los lenguajes siguientes:

1. $\text{NOPREFIJO}(L) = \{u \in L \mid \text{ningún prefijo propio de } u \text{ pertenece a } L\}$,

Como L es regular, existe un AFD $M = (Q, A, \delta, q_0, F)$ tal que $L = \mathcal{L}(M)$. Construimos un AFD $M' = (Q \cup \{E\}, A, \delta', q_0, F)$, donde E es un estado de error ($E \notin Q$) y δ' es:

$$\begin{cases} \delta'(q, a) = \delta(q, a) & \forall q \in Q \setminus F, a \in A \\ \delta'(q, a) = E & \forall q \in F, a \in A \\ \delta'(E, a) = E & \forall a \in A \end{cases}$$

Demostramos mediante doble inclusión que $\text{NOPREFIJO}(L) = \mathcal{L}(M')$.

\subseteq) Sea $u \in \text{NOPREFIJO}(L)$. Entonces, por definición de $\text{NOPREFIJO}(L)$, $u \in L$. Por tanto, $\exists q \in F$ tal que $\delta^*(q_0, u) = q$. Para ver que $u \in \mathcal{L}(M')$, basta ver que $(\delta')^*(q_0, u) \in F$.

Como u no tiene prefijos propios en L , entonces $\delta^*(q_0, u') \notin F$ para todo prefijo propio u' de u ; es decir, en los pasos de cálculo desde q_0 hasta $\delta^*(q_0, u)$ no se pasa por ningún estado final. Por tanto, como en esos casos $\delta' = \delta$, entonces $\delta^*(q_0, u) = q \in F$, por lo que $u \in \mathcal{L}(M')$.

\supseteq) Sea $u \in \mathcal{L}(M')$. En primer lugar, tenemos que $(\delta')^*(q_0, u) \in F$. Veamos ahora que los pasos de cálculo desde q_0 hasta $(\delta')^*(q_0, u)$ leyendo u no son ninguno finales.

Si alguno de ellos fuese final (si u tuviese algún prefijo propio $v \in L$), entonces desde él pasaríamos a E , y de este estado no final no saldríamos, llegando a contradicción. Por tanto, u no tiene prefijos propios pertenecientes a L . Además, como en estos casos $\delta' = \delta$, tenemos que $\delta^*(q_0, u) \in F$, luego $u \in L$. De esta forma, $u \in \text{NOPREFIJO}(L)$.

2. $\text{NOEXTENSION}(L) = \{u \in L \mid u \text{ no es un prefijo propio de ninguna palabra de } L\}$.

Como L es regular, existe un AFD $M = (Q, A, \delta, q_0, F)$ tal que $L = \mathcal{L}(M)$.

Construimos un AFD $M' = (Q, A, \delta, q_0, F')$, donde:

$$F' = \{q \in F \mid \delta^*(q, u) \notin F, \forall u \in A^*\}$$

Demostramos mediante doble inclusión que $\text{NOEXTENSION}(L) = \mathcal{L}(M')$.

\subseteq) Sea $u \in \text{NOEXTENSION}(L)$. Entonces, por definición de $\text{NOEXTENSION}(L)$, $u \in L$. Por tanto, $\exists q \in F$ tal que $\delta^*(q_0, u) = q$. Para ver que $u \in \mathcal{L}(M')$, basta ver que $q \in F'$.

Supongamos por reducción al absurdo $q \notin F'$. Entonces, $\exists v \in A^*$ tal que $\delta^*(q, v) \in F$. Pero entonces, $\delta^*(q_0, uv) = \delta^*(\delta^*(q_0, u), v) = \delta^*(q, v) \in F$, por lo que $uv \in L$ y, por tanto, u es prefijo propio de uv , lo cual es una contradicción. Por tanto, $q \in F'$ y, por tanto, $u \in \mathcal{L}(M')$.

\supseteq) Sea $u \in \mathcal{L}(M')$. En primer lugar, tenemos que $\delta^*(q_0, u) = q \in F' \subset F$, luego $u \in L$. Veamos ahora que u no es prefijo propio de ninguna palabra de L .

Supongamos por reducción al absurdo que u es prefijo propio de alguna palabra de L . Entonces, $\exists v \in A^* \setminus \{\varepsilon\}$ tal que $uv \in L$. Por tanto, $\delta^*(q_0, uv) \in F$. Pero entonces, $\delta^*(q_0, uv) = \delta^*(\delta^*(q_0, u), v) = \delta^*(q, v) \in F$. No obstante, hemos demostrado entonces que $q \notin F'$, lo cual es una contradicción. Por tanto, u no es prefijo propio de ninguna palabra de L y, por tanto, $u \in \text{NOEXTENSION}(L)$.

Ejercicio 1.2.24. Si $L \subseteq A^*$, define la relación \equiv en A^* como sigue: si $u, v \in A^*$, entonces $u \equiv v$ si y solo si para toda $z \in A^*$, tenemos que $(uz \in L \Leftrightarrow vz \in L)$.

1. Demostrar que \equiv es una relación de equivalencia.

Veamos las tres propiedades de las relaciones de equivalencia:

- Reflexiva: Sea $u \in A^*$. Entonces, para todo $z \in A^*$, tenemos trivialmente que $(uz \in L \Leftrightarrow uz \in L)$. Por tanto, $u \equiv u$.
- Simétrica: Sean $u, v \in A^*$ tales que $u \equiv v$. Entonces, para todo $z \in A^*$, tenemos que $(uz \in L \Leftrightarrow vz \in L)$. Por tanto, para todo $z \in A^*$, tenemos que $(vz \in L \Leftrightarrow uz \in L)$, lo cual implica que $v \equiv u$.
- Transitiva: Sean $u, v, w \in A^*$ tales que $u \equiv v$ y $v \equiv w$. Entonces, para todo $z \in A^*$, tenemos que $(uz \in L \Leftrightarrow vz \in L)$ y $(vz \in L \Leftrightarrow wz \in L)$. Por tanto, para todo $z \in A^*$, tenemos que $(uz \in L \Leftrightarrow wz \in L)$, lo cual implica que $u \equiv w$.

Tenemos por tanto que \equiv es una relación de equivalencia.

2. Calcular las clases de equivalencia de $L = \{a^i b^i \mid i \geq 0\}$.

En este caso, $A = \{a, b\}$. La primera clase de equivalencia que encontramos es las palabras que, le añadamos al final lo que le añadamos, no pertenecen al lenguaje. Es decir:

$$[u \in A^* \mid \text{en } u \text{ hay una } a \text{ después de una } b] \quad \vee \quad u = a^i b^j, \quad j > i]$$

Por comodidad, ya que $b \notin L$, considerando este representante de clase de equivalencia, notaremos a esta clase de equivalencia con $[b]$. Además, para cada $k \in \mathbb{N} \cup \{0\}$, tenemos:

$$[a_k] =: \{a^{k+j} b^j \mid j \in \mathbb{N} \cup \{0\}\}$$

Veamos en primer lugar que no hay más clases de equivalencia. Dado $u \in A^*$, si u tiene una a después de una b , entonces $u \in [b]$. En caso contrario, tenemos $u = a^i b^j$ con $i, j \in \mathbb{N} \cup \{0\}$.

- Si $j > i$, entonces $u \in [b]$.
- Si $j \leq i$, entonces $u \in [a_{i-j}]$.

Por tanto, tenemos que no hay más clases de equivalencia. Veamos ahora que estas clases de equivalencia son disjuntas.

- Sea $u \in [b]$. Si u tiene una a después de una b , entonces de forma directa $\nexists k \in \mathbb{N} \cup \{0\}$ tal que $u \in [a_k]$, ya que las palabras de estas clase de equivalencia son casos particulares de $a^* b^*$. Por otro lado, si $u = a^i b^j$ con $j > i$, entonces para que sea de la forma $a^{k+j} b^j$ es necesario que $k + j = i$, por lo que $k = i - j < 0$, luego $\nexists k \in \mathbb{N} \cup \{0\}$ tal que $u \in [a_k]$.

En conclusión, vemos que $[b] \cap [a_k] = \emptyset$ para todo $k \in \mathbb{N} \cup \{0\}$.

- Sean $k_1, k_2 \in \mathbb{N} \cup \{0\}$ tales que $k_1 \neq k_2$. Supongamos que $[a_{k_1}] \cap [a_{k_2}] \neq \emptyset$. Entonces, $\exists u \in A^*$ tal que $u \in [a_{k_1}] \cap [a_{k_2}]$. Por tanto, tenemos que $u = a^{k_1+j_1} b^{j_1} = a^{k_2+j_2} b^{j_2}$ con $j_1, j_2 \in \mathbb{N} \cup \{0\}$. Igualando las potencias de a y b , tenemos que $k_1 + j_1 = k_2 + j_2$ y $j_1 = j_2$. Por tanto, $k_1 = k_2$, lo cual es una contradicción. Por tanto, $[a_{k_1}] \cap [a_{k_2}] = \emptyset$ para todo $k_1, k_2 \in \mathbb{N} \cup \{0\}$ tales que $k_1 \neq k_2$.

Por tanto, vemos que las clases de equivalencia de L son $[b]$ y $[a_k]$ para todo $k \in \mathbb{N} \cup \{0\}$. Notemos que:

- $[b]$ es la clase de equivalencia de las palabras que, le añadamos al final lo que le añadamos, no pertenecen al lenguaje. Para cualquier par $u, v \in [b]$, tenemos que, para todo $z \in A^*$, $uz \in L \Leftrightarrow vz \in L$ es cierto por vacuidad, puesto que en ambos casos $uz, vz \notin L$.
- $[a_0]$ es la clase de equivalencia de las palabras que pertenecen al lenguaje. Para cualquier par $u, v \in [a_0]$, tenemos que, para todo $z \in A^*$, $uz \in L \Leftrightarrow vz \in L$ es cierto. Tomando $z = \varepsilon$, tenemos que $u = uz, v = vz \in L$; mientras que si $z \neq \varepsilon$, entonces $uz, vz \notin L$.

- $[a_k]$ para $k \neq 0$ es la clase de equivalencia de las palabras que tan solo pertenecen al lenguaje al concatenarles b^k . Para cualquier par $u, v \in [a_k]$, tenemos que, para todo $z \in A^*$, $uz \in L \Leftrightarrow vz \in L$ es cierto. Tomando $z = b^k$, tenemos que $ub^k, vb^k \in L$; mientras que si $z \neq b^k$, entonces $uz, vz \notin L$.

3. Calcular las clases de equivalencia de $L = \{a^i b^j \mid i, j \geq 0\}$.

La primera clase de equivalencia que encontramos es las palabras que, le añadamos al final lo que le añadamos, no pertenecen al lenguaje. Es decir:

$$[u \in A^* \mid \text{en } u \text{ hay una } a \text{ después de una } b]$$

Por comodidad, ya que $ba \notin L$, considerando este representante de clase de equivalencia, notaremos a esta clase de equivalencia con $[ba]$. Además, tenemos dos clases de equivalencia más:

$$[a] = \{a^i \mid i \in \mathbb{N} \cup \{0\}\}$$

$$[ab] = \{a^i b^j \mid i \in \mathbb{N} \cup \{0\}, j \in \mathbb{N}\}$$

Veamos en primer lugar que no hay más clases de equivalencia. Dado $u \in A^*$, si u tiene una a después de una b , entonces $u \in [ba]$. En caso contrario, tenemos $u = a^i b^j$ con $i, j \in \mathbb{N} \cup \{0\}$.

- Si $j = 0$, entonces $u \in [a]$.
- Si $j > 0$, entonces $u \in [ab]$.

Por tanto, tenemos que no hay más clases de equivalencia. Veamos ahora que estas clases de equivalencia son disjuntas.

- Sea $u \in [ba]$. Como u tiene una a después de una b , entonces de forma directa tenemos que $u \notin [a], [ab]$. En conclusión, vemos que $[ba] \cap [a] = [ba] \cap [ab] = \emptyset$.
- Sea $u \in [ab]$. Como $u = a^i b^j$ con $i, j \in \mathbb{N} \cup \{0\}$ con $j \neq 0$, entonces $u \notin [a]$. Por tanto, vemos que $[ab] \cap [a] = \emptyset$.

Por tanto, vemos que las clases de equivalencia de L son $[ba]$, $[a]$ y $[ab]$; y estas son disjuntas.

4. Demostrar que L es aceptado por un autómata finito determinista si y solo si el número de clases de equivalencia es finito.

Demostramos mediante doble inclusión.

\Rightarrow) Supongamos que L es aceptado por un autómata finito determinista. Sea $M = (Q, A, \delta, q_0, F)$ su AFD minimal que acepta L . Supongamos $u, v \in A^*$ tales que $u \equiv v$. Sean:

$$q_u := \delta^*(q_0, u) \in F,$$

$$q_v := \delta^*(q_0, v) \in F.$$

Veamos ahora que q_u, q_v son indistinguibles, es decir, $q_u = q_v$.

- Para todo $z \in A^*$, como $u \equiv v$, se tiene que:

$$uz \in L \Leftrightarrow vz \in L.$$

Equivalentemente, tenemos que:

$$\delta^*(\delta^*(q_0, u), z) \in F \iff \delta^*(\delta^*(q_0, v), z) \in F$$

Es decir:

$$\delta^*(q_u, z) \in F \iff \delta^*(q_v, z) \in F$$

Por tanto, q_u y q_v son indistinguibles; y como el autómata es minimal, $q_u = q_v$.

Por tanto, hemos demostrado que si $u \equiv v$, entonces $\delta^*(q_0, u) = \delta^*(q_0, v)$, por lo que el número de clases de equivalencia es menor o igual que el número de estados de Q . Como Q es finito, el número de clases de equivalencia también lo es finito.

\Leftarrow) Supongamos que el número de clases de equivalencia es finito. Sea el autómata $M = (Q, A, \delta, q_0, F)$, donde:

- Q es el conjunto de clases de equivalencia de L ,
- $q_0 = [\varepsilon]$,
- $\delta([u], a) = [ua]$. Veamos que está bien definida.
Sea $u, v \in A^*$ tales que $u \equiv v$, y veamos que, para todo $a \in A$, $ua \equiv va$. Como $u \equiv v$, para todo $z \in A^*$, tenemos que:

$$uz \in L \Leftrightarrow vz \in L.$$

Por tanto, tomando $z = az'$, con $z' \in L$, tenemos que:

$$uaz' \in L \Leftrightarrow vaz' \in L.$$

Es decir, $ua \equiv va$. Por tanto, δ está bien definida.

- $F = \{[u] \in Q \mid u \in L\}$.
Para ver que F está bien definida, veamos que, si $u \equiv v$, entonces $u \in L \iff v \in L$. Esto es directo tomando $z = \varepsilon$.

Veamos ahora que $L = \mathcal{L}(M)$.

$$\begin{aligned} u \in \mathcal{L}(M) &\iff \delta^*(q_0, u) \in F \iff \delta^*([\varepsilon], u) \in F \iff [\varepsilon u] \in F \iff [u] \in F \iff \\ &\iff u \in L \end{aligned}$$

5. ¿Qué relación existe entre el número de clases de equivalencia y el autómata finito minimal que acepta L ?

Veamos que el autómata descrito en el apartado anterior es minimal. En primer lugar, hemos de demostrar que no tiene estados inaccesibles.

- Sea $q \in Q$ una clase de equivalencia de L . Tomando un representante $u \in q$, tenemos que $\delta^*(q_0, u) = q$. Por tanto, q es accesible.

Veamos ahora que no tiene estados indistinguibles.

- Sean $q_1, q_2 \in Q$ clases de equivalencia distintas de L . Tomando representantes $u_1 \in q_1, u_2 \in q_2$, tenemos que:

$$\delta^*(q_0, u_1) = [u_1] = q_1,$$

$$\delta^*(q_0, u_2) = [u_2] = q_2.$$

Entonces, como son clases de equivalencia distintas, $u_1 \not\equiv u_2$, lo cual implica que $\exists z \in A^*$ tal que $u_1 z \in L$ y $u_2 z \notin L$ o viceversa. Supongamos sin pérdida de generalidad el primer caso, luego:

$$\exists z \in A^* \mid u_1 z \in L \quad \wedge \quad u_2 z \notin L \iff \delta^*(q_0, u_1 z) \in F \quad \wedge \quad \delta^*(q_0, u_2 z) \notin F$$

Por tanto, q_1 y q_2 no son indistinguibles.

Por tanto, hemos visto que todos los estados de Q son accesibles y no son indistinguibles, por lo que el autómata M del ejercicio anterior es minimal. Por tanto, la relación es que el número de clases de equivalencia es igual al número de estados del autómata finito minimal que acepta L .

Ejercicio 1.2.25. Dada una palabra $u = a_1 \cdots a_n \in A^*$, se llama $\text{Per}(u)$ al conjunto

$$\{a_{\sigma(1)}, \dots, a_{\sigma(n)} \mid \sigma \text{ es una permutación de } \{1, \dots, n\}\}.$$

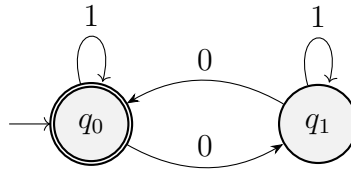
Dado un lenguaje L , se llama $\text{Per}(L) = \bigcup_{u \in L} \text{Per}(u)$. Dar expresiones regulares y autómatas minimales para $\text{Per}(L)$ en los siguientes casos:

1. $L = (00 + 1)^*$,

Tenemos que:

$$\text{Per}(L) = \{u \in A^* \mid n_0(u) \text{ es par}\}$$

Su autómata finito minimal es:



Este es de forma directa minimal, puesto que sus dos estados son distinguibles al ser uno final y el otro no. Para obtener la expresión regular, resolvemos el sistema de ecuaciones:

$$q_0 = 1q_0 + 0q_1 + \varepsilon$$

$$q_1 = 1q_1 + 0q_0$$

De la segunda ecuación, obtenemos $q_1 = 1^*0q_0$. Sustituyendo en la primera, obtenemos:

$$q_0 = 1q_0 + 0(1^*0q_0) + \varepsilon$$

$$q_0 = 1q_0 + 01^*0q_0 + \varepsilon$$

$$q_0 = (1 + 01^*0)q_0 + \varepsilon$$

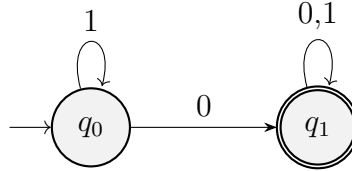
$$q_0 = (1 + 01^*0)^*$$

De forma directa, podríamos haber obtenido la siguiente expresión regular:

$$1^*(01^*01^*)^*$$

2. $L = (0 + 1)^*0$,

Tenemos que $\text{Per}(L) = \{u \in A^* \mid n_0(u) > 0\}$. Su autómata finito minimal es:



Este es de forma directa minimal, puesto que sus dos estados son distinguibles al ser uno final y el otro no. Para obtener la expresión regular, resolvemos el sistema de ecuaciones:

$$\begin{aligned} q_0 &= 1q_0 + 0q_1 \\ q_1 &= (0 + 1)q_1 + \varepsilon \end{aligned}$$

Tenemos por tanto que $q_1 = (0 + 1)^*$, y sustituyendo en la primera ecuación, obtenemos:

$$q_0 = 1q_0 + 0(0 + 1)^* = 1^*0(0 + 1)^*$$

3. $L = (01)^*$.

Tenemos que $\text{Per}(L) = \{u \in A^* \mid n_0(u) = n_1(u)\}$. En este caso, veamos que el lenguaje no es regular usando el Lema de Bombeo. Para todo $n \in \mathbb{N}$, consideramos la palabra $z = 0^n 1^n$, que cumple $|z| \geq n$. Si consideramos una descomposición $z = uvw$ con $|uv| \leq n$ y $|v| \geq 1$, tenemos que:

$$u = 0^k, \quad v = 0^l, \quad w = 0^{n-k-l} 1^n, \quad \text{con } k + l \leq n, l \geq 1$$

Entonces, tomando $i = 2$, tenemos que $uv^2w = 0^{n+l} 1^n \notin L$. Por tanto, L no es regular.

¿Es posible que, siendo L regular, $\text{Per}(L)$ no lo sea?

Como hemos visto en el apartado anterior, el lenguaje $L = (01)^*$ es regular, pero su permutación no lo es. Por tanto, es posible que, siendo L regular, $\text{Per}(L)$ no lo sea.

1.2.1. Preguntas Tipo Test

Se pide discutir la veracidad o falsedad de las siguientes cuestiones.

1. Si r y s son expresiones regulares, siempre se verifica que $(rs)^* = r^*s^*$.

Falso, si $r = 0$ y $s = 1$, tenemos que:

- La expresión regular $(01)^*$ está asociada al lenguaje $L_1 = \{(01)^i \mid i \in \mathbb{N}\}$.
- La expresión regular 0^*1^* está asociada al lenguaje $L_2 = \{0^i1^j \mid i, j \in \mathbb{N}\}$.

Y los lenguajes no son iguales, ya que $011 \in L_2 \setminus L_1$.

2. Si r y s son expresiones regulares, siempre se verifica que $(r + s)^* = r^* + s^*$.

Falso, si $r = 0$ y $s = 1$, tenemos que:

- La expresión regular $(0 + 1)^*$ está asociada al lenguaje $L_1 = \{0, 1\}^*$.
- La expresión regular $0^* + 1^*$ está asociada al lenguaje $L_2 = \{0\}^* \cup \{1\}^*$.

Y como $010 \in L_1 \setminus L_2$, no son iguales.

3. Si r_1 y r_2 son expresiones regulares tales que su lenguaje asociado contiene la palabra vacía, entonces $(r_1 r_2)^* = (r_2 r_1)^*$

Verdadero, sean R_1 y R_2 los lenguajes asociados a r_1 y a r_2 respectivamente, tratamos de comprobar si

$$(R_1 R_2)^* = (R_2 R_1)^*$$

con la condición de que $\varepsilon \in R_1 \cap R_2$.

- Sea $u \in (R_1 R_2)^*$, entonces u será de la forma

$$u = v_1 w_1 v_2 w_2 \dots v_n w_n \quad v_i \in R_1, \quad w_i \in R_2 \quad \forall i \in \{1, \dots, n\}$$

y podemos reescribir u como:

$$u = \varepsilon v_1 w_1 v_2 w_2 \dots v_n w_n \varepsilon \in (R_2 R_1)^*$$

- De forma análoga, si $u \in (R_2 R_1)^*$, podemos llegar a que $u \in (R_1 R_2)^*$.

4. Si r y s son expresiones regulares, siempre se verifica que $(r + \varepsilon)^* = r^*$.

Verdadero, sea R el lenguaje asociado a r , tenemos que

$$(R \cup \{\varepsilon\})^* = R^*$$

con lo que $(r + \varepsilon)^* = r^*$.

5. Si r y s son expresiones regulares, siempre se verifica que $r(r + s)^* = (r + s)^* r$.

Falso, sean $r = 0$ y $s = 1$, tenemos que:

- $0(0 + 1)^*$ es la expresión regular asociada al lenguaje $L_1 = \{0u \mid u \in \{0, 1\}^*\}$.
- $(0 + 1)^*0$ es la expresión regular asociada al lenguaje $L_2 = \{u0 \mid u \in \{0, 1\}^*\}$.

Como $0111 \in L_1 \setminus L_2$, los lenguajes no son iguales.

6. Si r_1 y r_2 son expresiones regulares, entonces $r_1^* r_2^* \subseteq (r_1 r_2)^*$, en el sentido de que los lenguajes asociados están incluidos.

Falso, si $r_1 = 1$ y $r_2 = 0$, estaríamos afirmando que $1^* 0^* \subseteq (10)^*$, lo cual es falso, ya que 110 es una palabra del lenguaje asociado a la primera expresión regular pero no al lenguaje asociado a la segunda.

7. Si r_1, r_2 y r_3 son expresiones regulares, entonces $(r_1 + r_2)^* r_3 = r_1^* r_3 + r_2^* r_3$.

Falso, si $r_1 = a$, $r_2 = b$ y $r_3 = c$, estaríamos afirmando que

$$(a + b)^* c = a^* c + b^* c$$

Pero abc es una palabra del lenguaje asociado a la primera expresión regular que no es del lenguaje asociado a la segunda expresión.

8. Si r_1 y r_2 son expresiones regulares entonces: $(r_1^* r_2^*)^* = (r_1 + r_2)^*$.

Verdadero, sean R_1 y R_2 los lenguajes asociados a r_1 y a r_2 respectivamente, tratamos de ver que

$$(R_1^* R_2^*)^* = (R_1 \cup R_2)^*$$

Lo cual puede demostrarse que es cierto.

9. Si r_1 y r_2 son expresiones regulares, entonces $(r_1 r_2)^* = (r_1 + r_2)^*$.

Falso, ya que si $r_1 = 0$ y $r_2 = 1$, estaríamos afirmando que

$$(01)^* = (0 + 1)^*$$

Pero 0 es una palabra del lenguaje asociado a la segunda expresión regular y no del lenguaje asociado a la primera.

10. Si r es una expresión regular, entonces $r^* r^* = r^*$.

Verdadero, si R es el lenguaje asociado, tenemos que ver que $R^* R^* = R^*$:

- Sea $u \in R^* R^*$, entonces u es de la forma:

$$u = v_1 v_2 \dots v_n v_{n+1} v_{n+2} \dots v_m \quad v_1 v_2 \dots v_n, v_{n+1} v_{n+2}, \dots v_m \in R^*$$

por lo que $v_i \in R \forall i \in \{1, \dots, m\}$ con lo que $u \in R^*$.

Si por contrario $u = \varepsilon$, entonces $u \in R^*$.

- Sea $u \in R^*$, entonces u será de la forma:

$$u = v_1 v_2 \dots v_n \quad v_i \in R \quad \forall i \in \{1, \dots, n\} \quad n > 1$$

por lo que podemos tomar $w = v_2 \dots v_n \in R^*$ y $v_1 \in R \subseteq R^*$, llegando a que

$$u = v_1 w \quad v_1, w \in R^* \implies u \in R^* R^*$$

- En el caso en el que $u \in R \subseteq R^*$, entonces $u = \varepsilon u$ con $\varepsilon \in R^*$.
- Además, si $u = \varepsilon$, entonces $u = \varepsilon \varepsilon$ con $\varepsilon \in R^*$.

11. Si r es una expresión regular, entonces $r\emptyset = r + \emptyset$.

Falso, sea R el lenguaje asociado a r , estaríamos afirmando que

$$\emptyset = R\emptyset = R \cup \emptyset = R$$

lo cual no es cierto, a no ser que $r = \emptyset$.

12. Si r es una expresión regular, entonces se verifica que $r^*\varepsilon = r^+\varepsilon$.

Verdadero, sea R el lenguaje asociado a r , entonces:

$$R^* \cup \{\varepsilon\} = R^* = R^+ \cup \{\varepsilon\}$$

13. Si r_1 y r_2 son expresiones regulares, entonces siempre $r_1(r_2r_1)^* = (r_1r_2)^*r_1$.

Verdadero, sean R_1 y R_2 los lenguajes asociados a r_1 y a r_2 respectivamente, entonces, tratamos de probar que

$$R_1(R_2R_1)^* = (R_1R_2)^*R_1$$

- Sea $u \in R_1(R_2R_1)^*$, entonces u es de la forma

$$u = v_0w_1v_1w_2v_2 \dots w_nv_n \quad v_i \in R_1 \quad w_i \in R_2$$

con lo que podemos tomar:

$$v = v_0w_1v_1w_2v_2 \dots w_n \in (R_1R_2)^*$$

y tenemos que $u = vv_n \in (R_1R_2)^*R_1$.

- Puede probarse de forma análoga que si $u \in (R_1R_2)^*R_1$, entonces $u \in R_1(R_2R_1)^*$.

14. Si r_1 y r_2 son expresiones regulares, siempre se verifica que $r_1(r_2r_1)^* = (r_1r_2)^*r_1$.

Verdadero, la pregunta es idéntica a la Pregunta 13.

15. Si r y s son expresiones regulares, entonces $(r^*s^*)^* = (r+s)^*$.

Verdadero, la pregunta es idéntica a la Pregunta 8.

16. Si r es una expresión regular, entonces $(rr)^* \subseteq r^*$.

Verdadero, sea R el lenguaje asociado a r , tratamos de ver que

$$(RR)^* \subseteq R^*$$

Sea $u \in (RR)^*$, entonces u es de la forma

$$u = v_1v'_1v_2v'_2 \dots v_nv'_n \quad v_i, v'_i \in R \quad \forall i \in \{1, \dots, n\}$$

por lo que $u \in R^*$.

17. Si r_1 y r_2 son expresiones regulares, tales que su lenguaje asociado contiene la palabra vacía, entonces $(r_1 r_2)^* = (r_1 + r_2)^*$.

Verdadero, puede probarse de forma similar a como lo hicimos en la pregunta 8.

18. Si r_1, r_2, r_3 son expresiones regulares, entonces $r_1(r_2^* + r_3^*) = r_1 r_2^* + r_1 r_3^*$.

Falso, podemos dar un contraejemplo de forma similar a como lo hicimos en la pregunta 7.

19. La demostración de que la clase de lenguajes aceptados por los autómatas no deterministas es la misma que la aceptada por los autómatas deterministas, se basa en dado un autómata no determinista construir uno determinista que, ante una palabra de entrada, explore todas las posibles opciones que puede seguir el no determinista.

Verdadero, así se comprueba que todo lenguaje aceptado por un autómata no determinista puede ser aceptado por uno determinista. No hace falta demostrarlo en la otra dirección, ya que un autómata determinista es a su vez no determinista.

20. Un autómata finito puede ser determinista y no-determinista a la vez.

Verdadero, ya que todos los autómatas deterministas son a su vez no deterministas.

21. Para transformar un autómata que acepta el lenguaje L en uno que acepte L^* , basta unir los estados finales con el inicial mediante transiciones nulas.

Falso, tenemos que hacer que el estado inicial sean también final, para que la palabra ε sea aceptada por el autómata, en caso de que $\varepsilon \notin L$.

22. Para pasar de un autómata que acepte el lenguaje asociado a r a uno que acepte r^* basta con unir con transiciones nulas sus estados finales con el estado inicial.

Falso, es la misma pregunta que la 21.

23. Existe un lenguaje reconocido por un AFD y no generado por una gramática independiente del contexto.

Falso, si un lenguaje es reconocido por un AFD, hemos visto en teoría que hay una gramática de tipo 3 que genera dicho lenguaje y como las gramáticas tipo 3 son a su vez independientes del contexto, el enunciado es falso.

24. Existen lenguajes aceptados por AFD que no pueden ser aceptados por AF no determinísticos.

Falso, el conjunto de lenguajes aceptados por un AFD coincide con el conjunto de lenguajes aceptados por AF no determinísticos, tal y como se ha visto en teoría.

25. La clausura de un lenguaje aceptado por un AFD puede ser representado con una expresión regular.

Verdadero, sea L un lenguaje aceptado por un AFD, conocemos por teoría que podemos encontrar una expresión regular r asociada a dicho lenguaje. Si ahora consideramos r^* , esta expresión regular estará asociada al lenguaje L^* .

26. Un lenguaje representado por una expresión regular siempre puede ser reconocido por un AF no determinista.

Verdadero, hemos visto en teoría que el conjunto de lenguajes representados por expresiones regulares coincide con el conjunto de lenguajes reconocidos por cualquier tipo de AF.

27. Todo lenguaje regular puede ser generado por una gramática libre de contexto.

Verdadero, ya que todo lenguaje regular puede ser generado por una gramática lineal por la derecha, que a su vez es independiente del contexto.

28. Un lenguaje con un número finito de palabras siempre puede ser reconocido por un AF no determinista.

Verdadero, sea $L = \{u_1, u_2, \dots, u_n\}$ un lenguaje finito de n palabras, entonces podemos considerar el autómata formado por un estado inicial q_0 y para cada palabra u_i con $i \in \{1, \dots, n\}$ de longitud $k = |u_i|$, añadimos k nuevos estados a los que llamaremos por ejemplo q_{ij} con $j \in \{1, \dots, k\}$. De esta forma, si

$$u_i = a_{i1}a_{i2} \dots a_{ik} \quad a_{ij} \in A \quad \forall j \in \{1, \dots, k\}$$

Entonces, añadimos las transiciones $\delta(q_{ij-1}, a_j) = q_{ij}$, entendiendo que $q_{i0} = q_0$ para cualquier $i \in \{1, \dots, n\}$ y consideraremos que q_{ik} es un estado final.

De manera más formal, dado un lenguaje finito $L = \{u_1, u_2, \dots, u_n\}$ sobre un alfabeto A , construimos el autómata $M = (Q, A, \delta, q_0, F)$ formado por los conjuntos:

$$\begin{aligned} Q &= \{q_0\} \cup \{q_{ij_i} \mid i \in \{1, \dots, n\}, j_i \in \{1, \dots, |u_i|\}\} \\ F &= \{q_{ij_i} \mid j_i = |u_i|\} \end{aligned}$$

donde la función de transición δ viene dada por:

$$\begin{aligned} \delta(q_0, a) &= \begin{cases} \{q_{i1}\} & \text{si } a = a_{i1} \\ \emptyset & \text{si } a \neq a_{i1} \end{cases} \quad \forall i \in \{1, \dots, n\} \\ \delta(q_{ij_i}, a) &= \begin{cases} \{q_{ij_i+1}\} & \text{si } a = a_{ij_i} \\ \emptyset & \text{si } a \neq a_{ij_i} \end{cases} \quad \forall i \in \{1, \dots, n\}, j_i \in \{1, \dots, |u_i|\} \end{aligned}$$

Es decir, para cada palabra creamos un camino desde el estado inicial a un estado final que se recorra leyendo dicha palabra.

Alternativamente, podríamos haber dicho que si $L = \{u_1, u_2, \dots, u_n\}$ es un lenguaje finito, entonces podemos considerar la gramática $G = (\{S\}, A, P, S)$ donde P es el conjunto:

$$P = \{S \rightarrow u_1, S \rightarrow u_2, \dots, S \rightarrow u_n\}$$

Y argumentar que dicha gramática puede pasarse a un AF no determinista por la teoría vista en el Tema 2.

29. Todo autómata finito determinista de n estados, cuyo alfabeto A contiene m símbolos debe tener $m \cdot n$ transiciones.

Verdadero, todo estado de un autómata finito determinista debe tener tantas transiciones como símbolos tenga su alfabeto A de entrada, en este caso, m . Como en este caso el autómata finito determinista tiene n estados, entonces el número de transiciones del autómata es:

$$\sum_{i=1}^n m = n \cdot m$$

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

No hemos visto autómatas con pila todavía, esta pregunta no es parte del Tema 2.

31. Si r_1 y r_2 son expresiones regulares, entonces siempre se tiene que $(r_1 + r_2)^* = (r_1^* r_2^*)^* r_1^*$.

Verdadero, puede razonarse buscando la igualdad entre los lenguajes que representan ambas expresiones.

32. Si un lenguaje es infinito no se puede encontrar una expresión regular que lo represente.

Falso, el lenguaje $L = \{1^n \mid n \in \mathbb{N}\}$ es infinito por ser \mathbb{N} infinito y puede representarse por la expresión regular 1^* .

33. Si r_1 y r_2 son expresiones regulares, entonces se verifica que $(r_1 + \varepsilon)^+ r_2^+ = r_1^+ (r_2 + \varepsilon)^+$.

Falso, si fuera cierto, entonces:

$$r_1^* r_2^+ = (r_1 + \varepsilon)^+ r_2^+ = r_1^+ (r_2 + \varepsilon)^+ = r_1^+ r_2^*$$

para cualesquiera r_1 y r_2 expresiones regulares.

Sin embargo, si tomamos $r_1 = 0$ y $r_2 = 1$, 1 es una palabra del lenguaje asociado a la expresión regular 0^*1^+ pero no lo es del lenguaje asociado a la expresión regular 0^+1^* , al no contener ningún “0”.

34. El conjunto de palabras sobre el alfabeto $\{0, 1\}$ tales que eliminando los tres últimos símbolos, en la palabra resultante no aparece el patrón 0011 es un lenguaje regular.

Verdadero, ya que podemos dar un AFD que reconozca dicho lenguaje:

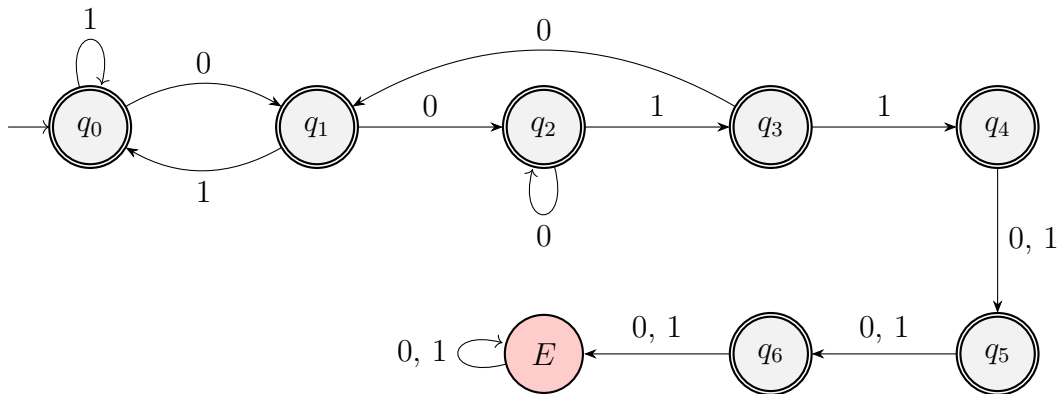


Figura 1.26: Autómata Finito Determinista para la pregunta 34.

35. El lenguaje formado por las cadenas sobre $\{0, 1\}$ que tienen un número impar de 0 y un número par de 1 no es regular.

Falso, ya que podemos dar una gramática $G = (\{A, B, C, D\}, \{0, 1\}, P, A)$ lineal por la derecha que genere dicho lenguaje. Si entendemos los estados $\{A, B, C, D\}$ como:

- A quiere decir que la palabra generada hasta el momento contiene un número par de 0s y de 1s.
- B quiere decir que la palabra generada hasta el momento contiene un número impar de 0s y par de 1s.
- C quiere decir que la palabra generada hasta el momento contiene un número par de 0s e impar de 1s.
- D quiere decir que la palabra generada hasta el momento contiene un número impar de 0s y de 1s.

Entonces, podemos dar las siguientes reglas de producción, que son las únicas reglas de producción que contiene P :

$$\begin{aligned}
 A &\rightarrow 0B \mid 1C \\
 B &\rightarrow \varepsilon \mid 0A \mid 1D \\
 C &\rightarrow 1A \mid 0D \\
 D &\rightarrow 1B \mid 0C
 \end{aligned}$$

Notemos que la variable inicial es A ya que ε contiene un número par de 0s y de 1s.

1.3. Propiedades de los Lenguajes Regulares

Ejercicio 1.3.1. Determinar si los siguientes lenguajes son regulares o libres de contexto. Justificar las respuestas.

1. $L_1 = \{0^i b^j \mid i = 2j \text{ ó } 2i = j\}$

Para todo $n \in \mathbb{N}$, consideramos la palabra $z = 0^{2n} b^n \in L_1$ con $|z| = 3n \geq n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0, b\}^*$, $|uv| \leq n$ y $|v| \geq 1$ debe cumplir que:

$$u = 0^k \quad v = 0^l \quad w = 0^{2n-k-l} b^n \quad \text{con } l, k \in \mathbb{N} \cup \{0\}, \quad l \geq 1, \quad k + l \leq n$$

Para $i = 2$, tenemos que $uv^i w = 0^{k+2l+2n-k-l} b^n = 0^{2n+l} b^n \notin L_1$, ya que, como $l \geq 1$:

$$2n + l \neq 2n \quad \text{y} \quad 2(2n + l) \neq n$$

Por tanto, por el recíproco del Lema de Bombeo, no es regular. Veamos ahora que sí es libre de contexto. Consideramos la gramática $G = (\{S, S_1, S_2\}, \{0, b\}, P, S)$ con P definido por:

$$\begin{aligned} S &\rightarrow S_1 \mid S_2 \\ S_1 &\rightarrow 00S_1 b \mid \varepsilon \\ S_2 &\rightarrow 0S_2 b b \mid \varepsilon \end{aligned}$$

Tenemos que G es una gramática libre de contexto tal que $\mathcal{L}(G) = L_1$, por lo que L_1 es libre de contexto.

2. $L_2 = \{uu^{-1} \mid u \in \{0, 1\}^*, |u| \leq 1000\}$

Consideramos el lenguaje auxiliar:

$$L'_2 = \{u \in \{0, 1\}^* \mid |u| \leq 2 \cdot 1000\}$$

Veamos que L'_2 es finito. Como el número de combinaciones de n elementos de $\{0, 1\}$ es 2^n , entonces el número de palabras de longitud menor o igual a $2 \cdot 1000$ es:

$$|L'_2| = \sum_{i=0}^{2 \cdot 1000} 2^i < \infty$$

Por tanto, como $L_2 \subset L'_2$ finito, tenemos que L_2 es finito y por tanto regular.

3. $L_3 = \{uu^{-1} \mid u \in \{0, 1\}^*, |u| \geq 1000\}$

Sabemos que el siguiente lenguaje es independiente del contexto:

$$L'_3 = \{uu^{-1} \mid u \in \{0, 1\}^*\}$$

Además, tenemos que $L'_3 = L_2 \cup L_3$. Supongamos que L_3 es regular. Entonces, como L_2 es regular, tendríamos que L'_3 es regular, lo cual es una contradicción.

Por tanto, L_3 no es regular. Para ver que es libre de contexto, consideramos la gramática $G = (V, \{0, 1\}, P, S)$ con:

$$V = \{S\} \cup \{A_i \mid i \in \{1, \dots, 1000\}\}$$

Tenemos que P está definido por:

$$\begin{aligned} S &\rightarrow 0A_10 \mid 1A_11, \\ A_i &\rightarrow 0A_{i+1}0 \mid 1A_{i+1}, \quad i \in \{1, \dots, 999\}, \\ A_{1000} &\rightarrow 0A_{1000}0 \mid 1A_{1000}1 \mid \varepsilon \end{aligned}$$

Notemos que, en A_i , ya hemos leído i caracteres de u (la palabra que forma la mitad del palíndromo). Una vez hemos llegado a A_{1000} , hemos leído 1000 caracteres de u . Por tanto, podemos añadir los que queramos sin restricción, y podemos también terminar. Como $L_3 = \mathcal{L}(G)$, tenemos que L_3 es independiente del contexto.

4. $L_4 = \{0^i 1^j 2^k \mid i = j \text{ ó } j = k\}$

Para todo $n \in \mathbb{N}$, consideramos la palabra $z = 0^n 1^n 2^{2n} \in L_4$ con $|z| = 3n \geq n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0, 1, 2\}^*$, $|uv| \leq n$ y $|v| \geq 1$ debe cumplir que:

$$u = 0^k \quad v = 0^l \quad w = 0^{n-k-l} 1^n 2^{2n} \quad \text{con } l, k \in \mathbb{N} \cup \{0\}, \quad l \geq 1, \quad k + l \leq n$$

Para $i = 2$, tenemos que $uv^i w = 0^{k+2l+n-k-l} 1^n 2^{2n} = 0^{n+l} 1^n 2^{2n} \notin L_4$, ya que, como $l \geq 1$:

$$n + l \neq n \quad \text{y} \quad n \neq 2n$$

Por tanto, por el recíproco del Lema de Bombeo, no es regular. Veamos ahora que sí es libre de contexto. Consideramos la gramática $G = (V, \{0, 1, 2\}, P, S)$ donde $V = \{S, S_1, S_2, A_0, A_2\}$ y P está definido por:

$$\begin{aligned} S &\rightarrow S_1 A_2 \mid A_0 S_2 \\ S_1 &\rightarrow 0 S_1 1 \mid \varepsilon \\ A_2 &\rightarrow 2 A_2 \mid \varepsilon \\ S_2 &\rightarrow 1 S_2 2 \mid \varepsilon \\ A_0 &\rightarrow 0 A_0 \mid \varepsilon \end{aligned}$$

Tenemos que G es una gramática libre de contexto tal que $\mathcal{L}(G) = L_4$, por lo que L_4 es libre de contexto.

Ejercicio 1.3.2. Determinar qué lenguajes son regulares o libres de contexto de los siguientes:

1. $\{u0u^{-1} \mid u \in \{0, 1\}^*\}$

Para todo $n \in \mathbb{N}$, consideramos la palabra $z = 0^n 10^n \in L$ con $|z| = 2n + 1 \geq n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0, 1\}^*$, $|uv| \leq n$ y $|v| \geq 1$ debe cumplir que:

$$u = 0^k \quad v = 0^l \quad w = 0^{n-k-l} 10^n \quad \text{con } l, k \in \mathbb{N} \cup \{0\}, \quad l \geq 1, \quad k + l \leq n$$

Para $i = 2$, tenemos que $uv^i w = 0^{k+2l+n-k-l}10^n = 0^{n+l}10^n \notin L$, ya que, como $l \geq 1$:

$$n + l \neq n$$

Por tanto, por el recíproco del Lema de Bombeo, no es regular. Veamos ahora que sí es libre de contexto. Consideramos la gramática $G = (\{S\}, \{0, 1\}, P, S)$, con P definido por:

$$S \rightarrow 0S01S1 \mid 0 \mid 1$$

Tenemos que G es una gramática libre de contexto tal que $\mathcal{L}(G) = L$, por lo que L es libre de contexto.

2. Números en binario que sean múltiplos de 4

Tenemos que todos los números en binario que son múltiplos de 4 terminan en 00, por lo que vienen dados por la siguiente expresión regular:

$$0^* + (1 + 0)^*00$$

Notemos que hemos incluido 0^* , porque el 0 también es múltiplo de 4. Por tanto, es regular.

3. Palabras de $\{0, 1\}^*$ que no contienen la subcadena 0110.

Notemos que podríamos dar un autómata que reconociese ese lenguaje, pero no es la opción más sencilla. Veamos en primer lugar que el lenguaje formado por las palabras que sí contienen la subcadena 0110 es regular dando una expresión regular asociada a él:

$$(0 + 1)^*0110(0 + 1)^*$$

Como el lenguaje descrito es su complementario y el complementario de un regular es regular, tenemos que el lenguaje dado es regular.

Ejercicio 1.3.3. Determinar qué lenguajes son regulares y qué lenguajes son libres de contexto entre los siguientes:

1. Conjunto de palabras sobre el alfabeto $\{0, 1\}$ en las que cada 1 va precedido por un número par de ceros.

Un reconocedor del lenguaje es el autómata de la figura 1.27, que tiene los siguientes estados:

- q_0 : Llevo un número par de ceros consecutivos, puedo leer un 1.
- q_1 : Llevo un número impar de ceros consecutivos, no puedo leer un 1.
- q_2 : Acabo de leer un 1.

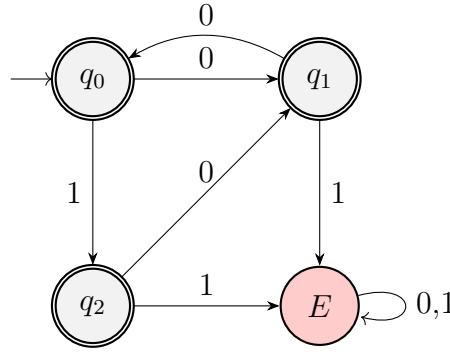


Figura 1.27: Autómata que reconoce el lenguaje del ejercicio 1.3.3.1.

Por tanto, como hemos dado un autómata que reconoce el lenguaje, es regular.

2. Conjunto $\{0^i 1^{2j} 0^{i+j} \mid i, j \geq 0\}$

Usaremos el Lema de Bombeo para demostrar que no es regular. Para todo $n \in \mathbb{N}$, consideramos la palabra $z = 0^n 1^{2n} 0^{2n}$ con $|z| = 5n \geq n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0, 1\}^*$, $|uv| \leq n$ y $|v| \geq 1$ debe cumplir que:

$$u = 0^k \quad v = 0^l \quad w = 0^{n-k-l} 1^{2n} 0^{2n} \quad \text{con } l, k \in \mathbb{N} \cup \{0\}, \quad l \geq 1, \quad k + l \leq n$$

Para $i = 2$, tenemos que $uv^i w = 0^{k+2l+n-k-l} 1^{2n} 0^{2n} = 0^{n+l} 1^{2n} 0^{2n} \notin L$, ya que, como $l \geq 1$:

$$2n \neq n + n + l$$

Por tanto, por el recíproco del Lema de Bombeo, no es regular. Veamos ahora que es libre de contexto. Consideramos la gramática $G = (\{S, X\}, \{0, 1\}, P, S)$, con P definido por:

$$\begin{aligned} S &\rightarrow 11S0 \mid X, \\ X &\rightarrow 0X0 \mid \varepsilon. \end{aligned}$$

Tenemos que G es una gramática libre de contexto tal que $\mathcal{L}(G) = L$, por lo que L es libre de contexto.

3. Conjunto $\{0^i 1^j 0^{i+j} \mid i, j \geq 0\}$

Usaremos el Lema de Bombeo para demostrar que no es regular. Para todo $n \in \mathbb{N}$, consideramos la palabra $z = 0^n 1^n 0^{n^2}$ con $|z| = n + n + n^2 \geq n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0, 1\}^*$, $|uv| \leq n$ y $|v| \geq 1$ debe cumplir que:

$$u = 0^k \quad v = 0^l \quad w = 0^{n-k-l} 1^n 0^{n^2} \quad \text{con } l, k \in \mathbb{N} \cup \{0\}, \quad l \geq 1, \quad k + l \leq n$$

Para $i = 2$, tenemos que $uv^i w = 0^{k+2l+n-k-l} 1^n 0^{n^2} = 0^{n+l} 1^n 0^{n^2} \notin L$, ya que, como $l \geq 1$:

$$(n + l) \cdot n = n^2 + nl \neq n^2$$

Por tanto, por el recíproco del Lema de Bombeo, no es regular. Además, este lenguaje no es libre de contexto (algo que aún no podemos demostrar).

Ejercicio 1.3.4. Determina si los siguientes lenguajes son regulares. Encuentra una gramática que los genere o un reconocedor que los acepte.

1. $L_1 = \{0^i 1^j \mid j < i\}$.

Usaremos el Lema de Bombeo para demostrar que no es regular. Para todo $n \in \mathbb{N}$, consideramos la palabra $z = 0^{n+1}1^n \in L_1$ con $|z| = 2n + 1 \geq n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0, 1\}^*$, $|uv| \leq n$ y $|v| \geq 1$ debe cumplir que:

$$u = 0^k \quad v = 0^l \quad w = 0^{n+1-k-l}1^n \quad \text{con } l, k \in \mathbb{N} \cup \{0\}, \quad l \geq 1, \quad k + l \leq n$$

Para $i = 0$, tenemos que $uv^i w = 0^{k+n+1-k-l}1^n = 0^{n+1-l}1^n \notin L_1$, ya que:

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

Pero esto es una contradicción, ya que $l \geq 1$. Por tanto, por el recíproco del Lema de Bombeo, no es regular. Veamos ahora que es libre de contexto. Consideramos la gramática $G = (\{S\}, \{0, 1\}, P, S)$, con P definido por:

$$\begin{aligned} S &\rightarrow 0S \mid 0S' \\ S' &\rightarrow 0S'1 \mid \varepsilon \end{aligned}$$

Tenemos que G es una gramática libre de contexto tal que $\mathcal{L}(G) = L_1$, por lo que L_1 es libre de contexto. Notemos que la producción $S \rightarrow 0S'$ fuerza a que haya al menos un 0 más que 1, y la producción $S' \rightarrow 0S'1$ permite que la diferencia no sea de una sola unidad, sino que pueda ser mayor.

2. $L_2 = \{001^i 0^j \mid i, j \geq 1\}$.

Tenemos que un reconocedor de L_2 es:

$$001^+ 0^+$$

Por tanto, tenemos que L_2 es regular.

3. $L_3 = \{010u \mid u \in \{0, 1\}^*, u \text{ no contiene la subcadena } 010\}$.

Sea $L' = \{u \in \{0, 1\}^* \mid u \text{ contiene la subcadena } 010\}$. Entonces sabemos que L' es regular con reconocedor:

$$(0 + 1)^* \mathbf{010} (0 + 1)^*$$

Por tanto, $\overline{L'}$ es regular, por lo que está asociado a una expresión regular, sea esta \tilde{r} . Entonces, L_3 es regular y está asociado a la expresión regular:

$$010\tilde{r}$$

Ejercicio 1.3.5. Sea el alfabeto $A = \{0, 1, +, =\}$, demostrar que el lenguaje

$$\text{ADD} = \{x = y + z \mid x, y, z \text{ son números en binario, y } x \text{ es la suma de } y \text{ y } z\}$$

no es regular.

Usaremos el Lema de Bombeo para demostrar que no es regular. Para todo $n \in \mathbb{N}$, consideramos la palabra $w = [1^n = 0 + 1^n]$, donde hemos empleado los corchetes para facilitar la notación (ya que el $=$ lo estamos usando para igualdades entre cadenas y entre números en binario). Tenemos que $|w| = n + 3 + n \geq n$. Toda descomposición $w = uvw$, con $u, v, w \in \{0, 1, +, =\}^*$, $|uv| \leq n$ y $|v| \geq 1$ debe cumplir que:

$$u = 1^k \quad v = 1^l \quad w = [1^{n-k-l} = 0 + 1^n] \quad \text{con } l, k \in \mathbb{N} \cup \{0\}, \quad l \geq 1, \quad k + l \leq n$$

Para $i = 2$, tenemos que $uv^i w = [1^{k+2l+n-k-l} = 0 + 1^n] = [1^{n+l} = 0 + 1^n] \notin \text{ADD}$, ya que, como $l \geq 1$, en números binarios:

$$1^{n+l} \neq 0 + 1^n = 1^n$$

Por tanto, por el recíproco del Lema de Bombeo, no es regular.

Ejercicio 1.3.6. Determinar si los siguientes lenguajes son regulares o no:

1. $L = \{uvu^{-1} \mid u, v \in \{0, 1\}^*\}$.

Veamos en primer lugar que $L = \{0, 1\}^*$ por doble inclusión:

⊂) Se tiene trivialmente que $L \subset \{0, 1\}^*$.

⊃) Sea $w \in \{0, 1\}^*$. Entonces, podemos tomar $u = \varepsilon$ y $v = w$ para obtener $w \in L$.

Por tanto, tenemos que L es regular, con reconocedor:

$$(0 + 1)^*$$

2. L es el lenguaje sobre el alfabeto $\{0, 1\}$ formado de las palabras de la forma $u0v$ donde u^{-1} es un prefijo de v .

Usaremos el Lema de Bombeo para demostrar que no es regular. Para todo $n \in \mathbb{N}$, consideramos la palabra $z = 1^n 0 1^n \in L$ con $|z| = 2n + 1 \geq n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0, 1\}^*$, $|uv| \leq n$ y $|v| \geq 1$ debe cumplir que:

$$u = 1^k \quad v = 1^l \quad w = 1^{n-k-l} 0 1^n \quad \text{con } l, k \in \mathbb{N} \cup \{0\}, \quad l \geq 1, \quad k + l \leq n$$

Para $i = 2$, tenemos que $uv^i w = 1^{k+2l+n-k-l} 0 1^n = 1^{n+l} 0 1^n \notin L$, ya que, como $l \geq 1$:

$$n + l \neq n \implies (1^{n+l})^{-1} = 1^{n+l} \neq 1^n$$

Por tanto, por el recíproco del Lema de Bombeo, no es regular.

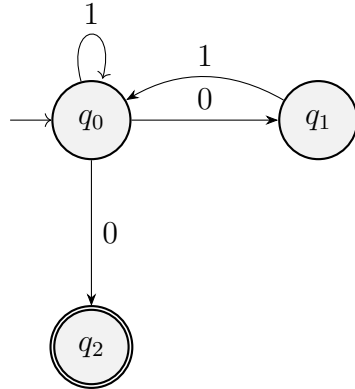
3. L es el lenguaje sobre el alfabeto $\{0, 1\}$ formado por las palabras en las que el tercer símbolo empezando por el final es un 1.

Este lenguaje es regular, con reconocedor:

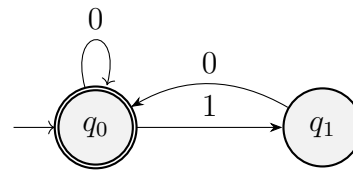
$$(0 + 1)^* \mathbf{1} (0 + 1) (0 + 1)$$

Ejercicio 1.3.7. Dar una expresión regular para la intersección de los lenguajes asociados a las expresiones regulares $(01 + 1)^*0$ y $(10 + 0)^*$. Se valorará que se construya el autómata que acepta la intersección de estos lenguajes, se minimice y, a partir del resultado, se construya la expresión regular.

Sea $r_1 = (01 + 1)^*0$ y $r_2 = (10 + 0)^*$. Construimos los AFND asociados a r_1 y r_2 , mostrados en las figuras 1.28a y 1.28b, respectivamente.



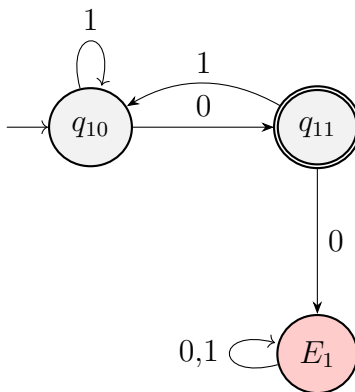
(a) AFND asociado a r_1 .



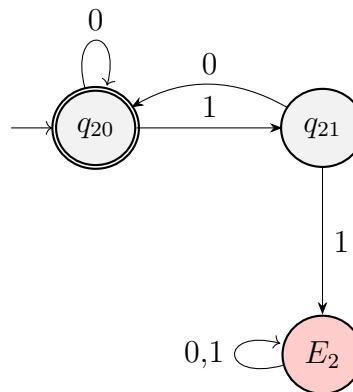
(b) AFND asociado a r_2 .

Figura 1.28: AFND asociados a las expresiones regulares r_1 y r_2 .

Para poder aplicar el algoritmo de intersección de autómatas, antes hemos de convertir los autómatas en AFD. Los AFD asociados a r_1 y r_2 son los de las figuras 1.29a y 1.29b, respectivamente.



(a) AFD asociado a r_1 .



(b) AFD asociado a r_2 .

Figura 1.29: AFD asociados a las expresiones regulares r_1 y r_2 .

Por tanto, el AFD que acepta la intersección de los lenguajes asociados a r_1 y r_2 es el de la figura 1.30.

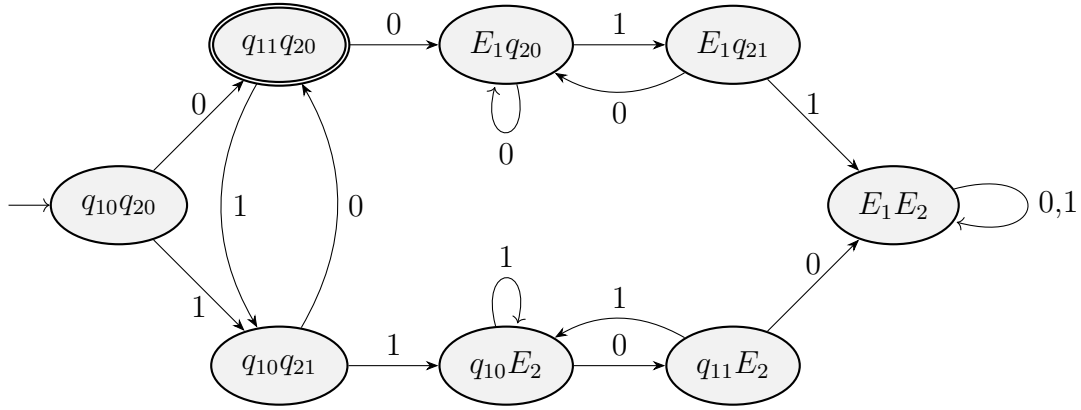


Figura 1.30: AFD que acepta la intersección de los lenguajes asociados a r_1 y r_2 .

Para minimizarlo, consideramos en primer lugar que los siguientes estados son indistinguibles:

$$q_E := \{E_1q_{20}, E_1q_{21}, q_{10}E_2, q_{11}E_2, E_1E_2\}$$

Estos son indistinguibles puesto que desde ninguno de ellos se puede llegar a un estado final. Por tanto, el AFD minimal es el de la figura 1.31.

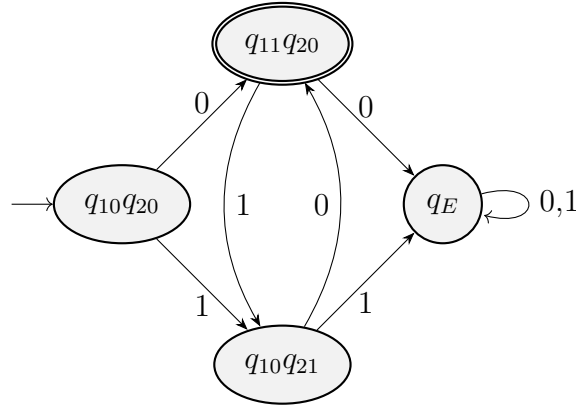


Figura 1.31: AFD minimal que acepta la intersección de los lenguajes asociados a r_1 y r_2 .

Tenemos que es minimal, puesto que todos los estados son alcanzables y no hay estados distinguibles:

- $q_{11}q_{20}$ es distinguible del resto de estados por ser el único estado final.
- q_E es distinguible de $q_{10}q_{20}$ y $q_{10}q_{21}$, ya que leyendo un 0:

$$\delta(q_E, 0) = q_E \notin F \quad \delta(q_{10}q_{20}, 0) = \delta(q_{10}q_{21}, 0) = q_{11}q_{20} \in F$$

- $q_{10}q_{20}$ y $q_{10}q_{21}$ son indistinguibles, ya que leyendo un 1:

$$\delta(q_{10}q_{20}, 1) = q_{10}q_{21} \notin F \quad \delta(q_{10}q_{21}, 1) = q_{11}q_{20} \in F$$

Por tanto, el AFD minimal que acepta la intersección de los lenguajes asociados a r_1 y r_2 es el de la figura 1.31. Para obtener la expresión regular asociada, resolvemos el sistema de ecuaciones:

$$\begin{aligned} q_{10}q_{20} &= 0q_{11}q_{20} + 1q_{10}q_{21} \\ q_{11}q_{20} &= 0q_E + 1q_{10}q_{21} + \varepsilon \\ q_{10}q_{21} &= 0q_{11}q_{20} + 1q_E \\ q_E &= 0q_E + 1q_E = (0 + 1)q_E \end{aligned}$$

Por el Lema de Arden, como $q_E = (0 + 1)q_E + \emptyset$, tenemos que $q_E = (0 + 1)^*\emptyset = \emptyset$. Sustituyendo en las ecuaciones anteriores, obtenemos:

$$\begin{aligned} q_{10}q_{20} &= 0q_{11}q_{20} + 1q_{10}q_{21} \\ q_{11}q_{20} &= 1q_{10}q_{21} + \varepsilon \\ q_{10}q_{21} &= 0q_{11}q_{20} \end{aligned}$$

Sustituyendo $q_{10}q_{21}$ en la segunda ecuación, obtenemos:

$$q_{11}q_{20} = 1q_{11}q_{20} + \varepsilon \implies q_{11}q_{20} = 10q_{11}q_{20} + \varepsilon = (10)^*\varepsilon = (10)^*$$

Sustituyendo ambos en la primera ecuación, obtenemos:

$$q_{10}q_{20} = 0(10)^* + 10q_{11}q_{20} = 0(10)^* + 10(10)^* = (0 + 10)(10)^*$$

Por tanto, la expresión regular asociada al AFD minimal de la figura 1.31 es

$$(0 + 10)(10)^*.$$

Ejercicio 1.3.8. Encontrar un AFD minimal para el lenguaje

$$(a + b)^*(aa + bb)(a + b)^*$$

Para ello, primero construimos el AFND asociado a la expresión regular, mostrado en la figura 1.32.

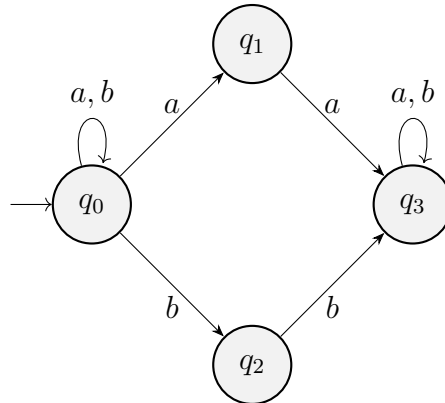


Figura 1.32: AFND asociado a la expresión regular $(a + b)^*(aa + bb)(a + b)^*$.

Convertimos el AFND en un AFD, mostrado en la figura 1.33.

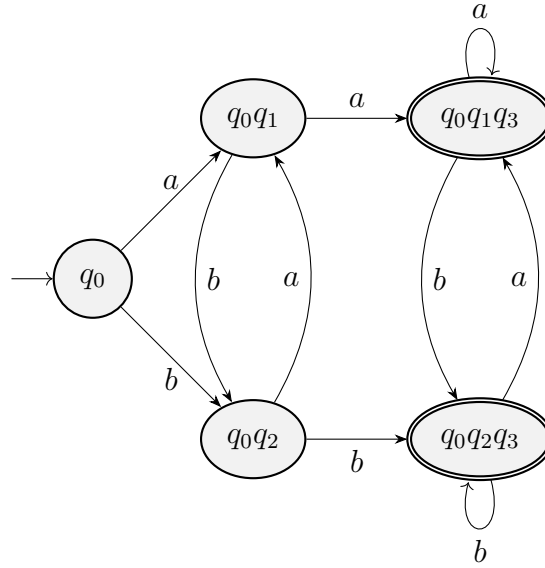


Figura 1.33: AFD asociado a la expresión regular $(a + b)^*(aa + bb)(a + b)^*$.

No obstante, este no es minimal. En primer lugar, vemos que los estados $q_0q_1q_3$ y $q_0q_2q_3$ son indistinguibles, ya que para cualquier palabra $w \in \{a, b\}^*$:

$$\delta(q_0q_1q_3, w) \in F \quad \delta(q_0q_2q_3, w) \in F$$

Por tanto, notemos $q_F = \{q_0q_1q_3, q_0q_2q_3\}$. El AFD minimal es el de la figura 1.34.

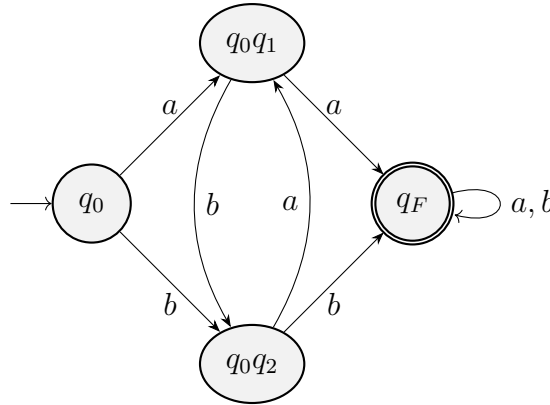


Figura 1.34: AFD minimal asociado a la expresión regular $(a + b)^*(aa + bb)(a + b)^*$.

- q_F es distinguible del resto de estados por ser el único estado final.
- q_0q_1 y q_0q_2 son indistinguibles, ya que leyendo un a :

$$\delta(q_0q_1, a) = q_F \in F \quad \delta(q_0q_2, a) = q_0q_1 \notin F$$

- q_0 y q_0q_1 son indistinguibles, ya que leyendo un a :

$$\delta(q_0, a) = q_0q_1 \notin F \quad \delta(q_0q_1, a) = q_F \in F$$

- q_0 y q_0q_2 son indistinguibles, ya que leyendo un b :

$$\delta(q_0, b) = q_0q_2 \notin F \quad \delta(q_0q_2, b) = q_F \in F$$

Por tanto, el AFD minimal es el de la figura 1.34.

Ejercicio 1.3.9. Para cada uno de los siguientes lenguajes regulares, encontrar el autómata minimal asociado, y a partir de dicho autómata minimal, determinar la gramática regular que genera el lenguaje:

1. a^+b^+

En primer lugar, construimos el AFD asociado al lenguaje, mostrado en la figura 1.35.

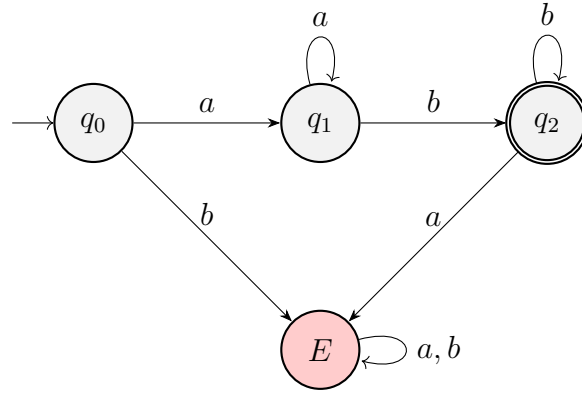


Figura 1.35: AFD asociado al lenguaje a^+b^+ del ejercicio 1.3.9.1.

Veamos que este es minimal:

- q_2 es distinguible del resto de estados por ser el único estado final.
- q_0 y q_1 son indistinguibles, ya que leyendo un b :

$$\delta(q_0, b) = E \notin F \quad \delta(q_1, b) = q_2 \in F$$

- q_0 y E son indistinguibles, ya que leyendo un ab :

$$\delta^*(q_0, ab) = q_2 \in F \quad \delta^*(E, ab) = E \notin F$$

- q_1 y E son indistinguibles, ya que leyendo un b :

$$\delta(q_1, b) = q_2 \in F \quad \delta(E, b) = E \notin F$$

Por tanto, el AFD minimal es el de la figura 1.35. La gramática regular que genera el lenguaje es $G = (\{q_0, q_1, q_2\}, \{a, b\}, P, \{q_0\})$ con P :

$$\begin{aligned} q_0 &\longrightarrow aq_1 \\ q_1 &\longrightarrow aq_1 \mid bq_2 \\ q_2 &\longrightarrow bq_2 \mid \varepsilon \end{aligned}$$

2. $a(a + b)^*b$

En primer lugar, construimos el AFND asociado al lenguaje, mostrado en la figura 1.36.

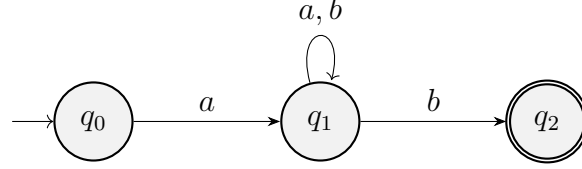


Figura 1.36: AFND asociado al lenguaje $a(a + b)^*b$ del ejercicio 1.3.9.2.

Convertimos el AFND en un AFD, mostrado en la figura 1.37.

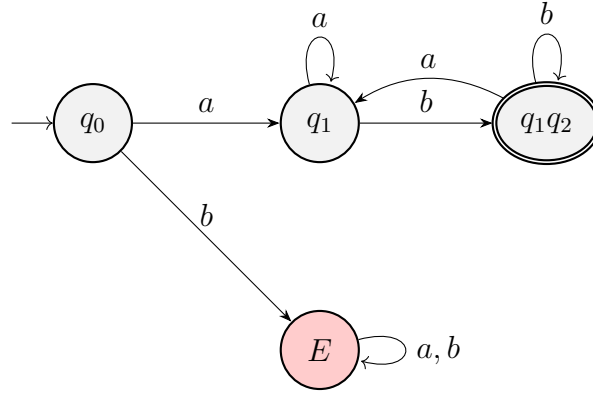


Figura 1.37: AFD asociado al lenguaje $a(a + b)^*b$ del ejercicio 1.3.9.2.

Veamos que este es minimal:

- q_1q_2 es distinguible del resto de estados por ser el único estado final.
- q_0 y q_1 son indistinguibles, ya que leyendo un b :

$$\delta(q_0, b) = E \notin F \quad \delta(q_1, b) = q_1q_2 \in F$$

- q_0 y E son indistinguibles, ya que leyendo un ab :

$$\delta^*(q_0, ab) = q_1q_2 \in F \quad \delta^*(E, ab) = E \notin F$$

- q_1 y E son indistinguibles, ya que leyendo un b :

$$\delta(q_1, b) = q_1q_2 \in F \quad \delta(E, b) = E \notin F$$

Ejercicio 1.3.10. Determinar autómatas minimales para los lenguajes $L(M_1) \cup L(M_2)$ y $L(M_1) \cap \overline{L(M_2)}$ donde,

1. $M_1 = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \delta_1, q_0, \{q_2\})$ donde

δ_1	q_0	q_1	q_2	q_3
a	q_1	q_1	q_3	q_3
b	q_2	q_1	q_1	q_3
c	q_3	q_3	q_0	q_3

2. $M_2 = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \delta_2, q_0, \{q_2\})$ donde

δ_2	q_0	q_1	q_2	q_3
a	q_1	q_1	q_3	q_3
b	q_1	q_2	q_2	q_3
c	q_3	q_3	q_0	q_3

En primer lugar, minimizamos M_1 y M_2 . La minimización de M_1 se muestra en la Tabla 1.1.

q_1	×		
q_2	×	×	
q_3	×	(q_0, q_3)	×
	q_0	q_1	q_2

Tabla 1.1: Tabla de minimización de M_1 .

Por tanto, $\{q_1, q_3\}$ son indistinguibles, por lo que el AFD minimal asociado a M_1 es $M_1^m = (\{q_0, q_1, q_2\}, \{a, b, c\}, \delta_1^m, q_0, \{q_2\})$ donde:

δ_1^m	q_0	q_1	q_2
a	q_1	q_1	q_1
b	q_2	q_1	q_1
c	q_1	q_1	q_0

La minimización de M_2 se muestra en la Tabla 1.2.

q_1	×		
q_2	×	×	
q_3	×	×	×
	q_0	q_1	q_2

Tabla 1.2: Tabla de minimización de M_2 .

Por tanto, el AFD minimal asociado a M_2 es $M_2^m = M_2$ minimal. A continuación, construimos los autómatas finitos deterministas asociados a $L(M_1) \cup L(M_2)$ y $L(M_1) \cap \overline{L(M_2)}$.

■ $L(M_1) \cup L(M_2)$

En primer lugar, construimos el AFD asociado a $L(M_1) \cup L(M_2)$. Sea este $M = (Q, \{a, b, c\}, \delta, (q_0, q'_0), F)$, donde:

- $Q = \{q_i q'_j \mid i = 0, 1, 2, 3 \text{ y } j = 0, 1, 2\}$.
- $F = \{(q_2, q'_j) \mid j = 0, 1, 2, 3\} \cup \{(q_i, q'_2) \mid i = 0, 1, 2\}$.
- δ viene dada por:

δ	(q_0, q'_0)	(q_0, q'_1)	(q_0, q'_2)	(q_0, q'_3)	(q_1, q'_0)	(q_1, q'_1)	(q_1, q'_2)	(q_1, q'_3)	(q_2, q'_0)	(q_2, q'_1)	(q_2, q'_2)	(q_2, q'_3)
a	$q_1 q'_1$	$q_1 q'_1$	$q_1 q'_3$	$q_1 q'_3$	$q_1 q'_1$	$q_1 q'_1$	$q_1 q'_3$	$q_1 q'_3$	$q_1 q'_1$	$q_1 q'_1$	$q_1 q'_3$	$q_1 q'_3$
b	$q_2 q'_1$	$q_2 q'_2$	$q_2 q'_2$	$q_2 q'_3$	$q_1 q'_1$	$q_1 q'_2$	$q_1 q'_2$	$q_1 q'_3$	$q_1 q'_1$	$q_1 q'_2$	$q_1 q'_2$	$q_1 q'_3$
c	$q_1 q'_3$	$q_1 q'_3$	$q_1 q'_0$	$q_1 q'_3$	$q_1 q'_3$	$q_1 q'_3$	$q_1 q'_0$	$q_1 q'_3$	$q_0 q'_3$	$q_0 q'_3$	$q_0 q'_0$	$q_0 q'_3$

Los estados accesibles son:

$$\{q_0q'_0, q_0q'_3, q_1q'_0, q_1q'_1, q_1q'_2, q_1q'_3, q_2q'_1, q_2q'_3\}$$

La minimización de M se muestra en la Tabla 1.3.

$q_0q'_3$						
$q_1q'_0$						
$q_1q'_1$						
$q_1q'_2$						
$q_1q'_3$						
$q_2q'_1$						
$q_2q'_3$						
	$q_0q'_0$	$q_0q'_3$	$q_1q'_0$	$q_1q'_1$	$q_1q'_2$	$q_1q'_3$

Tabla 1.3: Tabla de minimización de M para $L(M_1) \cup L(M_2)$.

Ejercicio 1.3.11. Dado el conjunto regular representado por la expresión regular $a^*b^* + b^*a^*$, construir un autómata finito determinístico minimal que lo acepte.

Ejercicio 1.3.12. Sean los lenguajes:

1. $L_1 = (01 + 1)^*00$
2. $L_2 = 01(01 + 1)^*$

construir un autómata finito determinístico minimal que acepte el lenguaje $L_1 \setminus L_2$, a partir de autómatas que acepten L_1 y L_2 .

Ejercicio 1.3.13. Dados los alfabetos $A = \{0, 1, 2, 3\}$ y $B = \{0, 1\}$ y el homomorfismo f de A^* en B^* dado por:

1. $f(0) = 00, f(1) = 01, f(2) = 10, f(3) = 11$

Sea L el conjunto de las palabras de B^* en las que el número de símbolos 0 es par y el de símbolos 1 no es múltiplo de 3. Construir un autómata finito determinista que acepte el lenguaje $f^{-1}(L)$.

Ejercicio 1.3.14. Determinar si las expresiones regulares siguientes representan el mismo lenguaje:

1. $(b + (c + a)a^*(b + c))^*(c + a)a^*$
2. $b^*(c + a)((b + c)b^*(c + a))^*a^*$
3. $b^*(c + a)(a^*(b + c)b^*(c + a))^*a^*$

Justificar la respuesta.

Ejercicio 1.3.15. Construir un autómata finito determinista minila que acepte el conjunto de palabras sobre el alfabeto $A = \{0, 1\}$ que representen números no divisibles por dos ni por tres.

Ejercicio 1.3.16. 1. Construye una gramática regular que genere el siguiente lenguaje:

$$L_1 = \{u \in \{0, 1\}^* \mid \text{el número de 1's y el número de 0's en } u \text{ es par} \}$$

2. Construye un autómata que reconozca el siguiente lenguaje:

$$L_2 = \{0^n 1^m \mid n \geq 1, m \geq 0, n \text{ múltiplo de } 3, m \text{ par} \}$$

3. Diseña el AFD mínimo que reconoce el lenguaje $(L_1 \cup L_2)$.

Ejercicio 1.3.17. Construir autómatas finitos para los siguientes lenguajes sobre el alfabeto $\{a, b, c\}$:

1. L_1 : palabras del lenguaje $(a + b)^*(b + c)^*$.
2. L_2 : palabras en las que nunca hay una 'a' posterior a una 'c'.
3. $(L_1 \setminus L_2) \cup (L_2 \setminus L_1)$

¿Qué podemos concluir sobre L_1 y L_2 ?

Ejercicio 1.3.18. Si $f : \{0, 1\}^* \rightarrow \{a, b, c\}^*$ es un homomorfismo dado por

$$f(0) = aab \quad f(1) = bbc$$

dar autómatas finitos deterministas minimales para los lenguajes L y $f^{-1}(L)$ donde $L \subseteq \{a, b, c\}^*$ es el lenguaje en el que el número de símbolos a no es múltiplo de 4.

Ejercicio 1.3.19. Si L_1 es el lenguaje asociado a la expresión regular $01(01 + 1)^*$ y L_2 el lenguaje asociado a la expresión $(1 + 10)^*01$, encontrar un autómata minimal que acepte el lenguaje $L_1 \setminus L_2$.

Ejercicio 1.3.20. Sean los alfabetos $A_1 = \{a, b, c, d\}$ y $A_2 = \{0, 1\}$ y el lenguaje $L \subseteq A_2^*$ dado por la expresión regular $(0 + 1)^*0(0 + 1)$, calcular una expresión regular para el lenguaje $f^{-1}(L)$ donde f es el homomorfismo entre A_1^* y A_2^* dado por

$$f(a) = 01 \quad f(b) = 1 \quad f(c) = 0 \quad f(d) = 00$$

Ejercicio 1.3.21. Dado el lenguaje L asociado a la expresión regular $(01 + 011)^*$ y el homomorfismo $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ dado por $f(0) = 01$, $f(1) = 1$, construir una expresión regular para el lenguaje $f^{-1}(L)$.

Ejercicio 1.3.22. Dar expresiones regulares para los siguientes lenguajes sobre el alfabeto $A_1 = \{0, 1, 2\}$:

1. L dado por el conjunto de palabras en las que cada 0 que no sea el último de la palabra va seguido por un 1 y cada 1 que no sea el último símbolo de la palabra va seguido por un 0.
2. Considera el homomorfismo de A_1 en $A_2 = \{0, 1\}$ dado por $f(0) = 001$, $f(1) = 100$, $f(2) = 0011$. Dar una expresión regular para $f(L)$.

3. Dar una expresión regular para LL^{-1} .

Ejercicio 1.3.23. Dados los lenguajes

$$L_1 = \{0^i 1^j \mid i \geq 1, j \text{ es par y } j \geq 2\}$$

y

$$L_2 = \{1^j 0^k \mid k \geq 1, j \text{ es impar y } j \geq 1\}$$

encuentra:

1. Una gramática regular que genere el lenguaje L_1 .
2. Una expresión regular que represente al lenguaje L_2 .
3. Un automata finito determinista que acepte las cadenas de la concatenación de los lenguajes $L_1 L_2$. Aplica el algoritmos para minimizar este autómata.