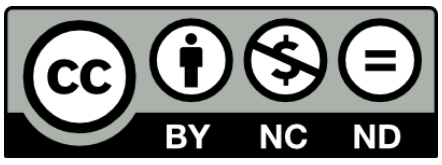
Modelos de Computación

Los Del DGIIM, losdeldgiim.github.io

Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada



Modelos de Computación

Los Del DGIIM, losdeldgiim.github.io

Arturo Olivares Martos

Granada, 2024-2025

Índice general

1. Relaciones de Problemas

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:

$$S \to XYX$$
$$X \to aX \mid bX \mid \varepsilon$$
$$Y \to bbb$$

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

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

$$S \Longrightarrow XYX \Longrightarrow XbbbX$$

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

$\supseteq$) 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 $X \to aX \mid bX \mid \varepsilon$. Por tanto, $w \in L$.

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

$$S \to aX$$
$$X \to aX \mid bX \mid \varepsilon$$

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

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

$$S \Longrightarrow aX \Longrightarrow au$$

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

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

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

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

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

$$S \to XaXaX$$
$$X \to aX \mid bX \mid \varepsilon$$

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

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

$$S \Longrightarrow XaXaX \Longrightarrow uavaw$$

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

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

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

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

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

$$S \to SS \mid XaXaX \mid \varepsilon$$
$$X \to bX \mid \varepsilon$$

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

$\subseteq$) 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, $z \in L \cdot \ldots \cdot L$ (n veces), por lo que existen $i_1, j_1, k_1, \ldots, i_n, j_n, k_n \in \mathbb{N} \cup \{0\}$ tales que $z = b^{i_1} a b^{j_1} a b^{k_1} \cdot \ldots \cdot b^{i_n} a b^{j_n} a b^{k_n}$. Veamos que $S \overset{*}{\Longrightarrow} z$:

- Para conseguir el número de a's deseado, empleamos la regla de producción $S \to SS$ 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 $X \to \varepsilon$.

$\supseteq$) 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 $S \to SS \to SXaXaX$ 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 $S \to \varepsilon$.
- Posteriormente, para cada X, tan solo podemos emplear la regla de producción $X \to bX \mid \varepsilon$ 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:

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

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

$\subseteq$) Sea $w \in L$. Entonces, $w = ua$ con $u \in \{abb, baab\}^*$. Veamos que $S \overset{*}{\Longrightarrow} w$. 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 $S \to abAS \to abbS$.
- Para producir $baab$, tenemos que $S \to abAS \to baabS$.

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

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

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

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

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):

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

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

$\subseteq$) 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 $\langle\text{número}\rangle \overset{*}{\Longrightarrow} w$:

- En primer lugar, aplicamos $|w| - 1$ veces la regla de producción $\langle\text{número}\rangle \to \langle\text{número}\rangle\langle\text{dígito}\rangle$ 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 $\langle\text{número}\rangle \to \langle\text{dígito}\rangle$ para reemplazar la última variable por un dígito, que será el primero del número formado.

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

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

$$\begin{aligned} \langle\text{número}\rangle &\Longrightarrow \langle\text{número}\rangle\langle\text{dígito}\rangle \overset{|w|-1 \text{ veces}}{\Longrightarrow} \\ &\Longrightarrow \langle\text{número}\rangle\langle\text{dígito}\rangle\langle\text{dígito}\rangle \overset{|w|-1 \text{ veces}}{\cdots} \langle\text{dígito}\rangle \Longrightarrow \\ &\Longrightarrow \langle\text{dígito}\rangle \overset{|w| \text{ veces}}{\cdots} \langle\text{dígito}\rangle \end{aligned}$$

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:

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

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

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

- En primer lugar, aplicamos $n-1$ veces la regla de producción $S \to aS$ para obtener $a^{n-1}S$,
$$S \overset{*}{\Longrightarrow} a^{n-1}S$$
- Para cambiar a la etapa de añadir b's, aplicamos la regla de producción $S \to aA$, obteniendo así $a^n A$,
- Después, aplicamos $m-1$ veces la regla de producción $A \to bA$ para obtener $a^n b^{m-1} A$.
- Para finalizar, aplicamos la regla de producción $A \to b$ para obtener $a^n b^m$.

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

$\supseteq$) 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:

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

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:

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

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:

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

   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:

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

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:

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

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 $Y \to \varepsilon$, 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:

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

   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:

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

   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:

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

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:

   $$\begin{aligned} V &= \{S\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{rcl} S & \to & aS \mid baS \mid b \mid \varepsilon \end{array} \right\} \end{aligned}$$

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:

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

   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:

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

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:

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

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

   Previamente a hacer este Ejercicio, se recomienda consultar el Ejercicio 1.1.5.

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

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

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:

$$\begin{aligned} V &= \{S, C_a, C_b, X\} \\ T &= \{a, b\} \\ S &= S \\ P &= \left\{ \begin{array}{rcl} S & \to & aC_aX \mid bC_bX \\ C_a & \to & a \mid XCX \\ C_b & \to & b \mid XDX \\ X & \to & a \mid b \end{array} \right\} \end{aligned}$$

   donde notemos que C_a fuerza a que la letra central sea una a, mientras que C_b fuerza a que la letra central sea una b.

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:

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

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

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

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:

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

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 $L^* = L$.

$$L = L^* \iff \begin{cases} \varepsilon \in L \\ \wedge \\ u, v \in L \implies uv \in L \end{cases}$$

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.

$\Longleftarrow$) 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 $L^* \subseteq L$.

$\Longrightarrow$) 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 $f : A^* \to B^*$, $g : A^* \to B^*$, 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.

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

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

$\Longrightarrow$) 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 $i \geqslant 1$, 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 $A = \{a_1, a_2, \ldots, a_n\}$, 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 buscamos una inyección de Γ en B^*. Dado un lenguaje $L \in \Gamma$, sea $L = \{l_1, l_2, \ldots, l_m\}$, con $m \in \mathbb{N}$ y $l_i \in A^*$ $\forall i \in \{1, \ldots, m\}$. Definimos la siguiente función:

$$\begin{array}{rccc} f: & \Gamma & \longrightarrow & B^* \\ & L & \longmapsto & zl_1zl_2 \ldots zl_mz \end{array}$$

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

$$zl_1zl_2 \ldots zl_kz = zl'_1zl'_2 \ldots zl'_{k'}z$$

Por ser ambas palabras iguales, tenemos que $k = k'$ y $l_i = l'_i$ $\forall i \in \{1, \ldots, k\}$, 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. $\{u \in \{0,1\}^* \mid |u| \leqslant 4\}$

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

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

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

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

   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:

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

   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:

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

   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:

$$
\begin{aligned}
V &= \{S\} \\
T &= \{a\} \\
S &= S \\
P &= \left\{ \begin{array}{lcl} S & \to & S \end{array} \right\}
\end{aligned}
$$

4. El lenguaje formado por los números naturales.

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

$$V = \{\langle\text{número no iniciado}\rangle, \langle\text{dígito no cero}\rangle, \langle\text{dígito}\rangle, \langle\text{número iniciado}\rangle\}$$
$$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$
$$S = \langle\text{número no iniciado}\rangle$$
$$P = \left\{ \begin{array}{rcl} \langle\text{número no iniciado}\rangle & \to & \langle\text{dígito no cero}\rangle \mid \langle\text{dígito no cero}\rangle\langle\text{número iniciado}\rangle \\ \langle\text{número iniciado}\rangle & \to & \langle\text{dígito}\rangle \mid \langle\text{dígito}\rangle\langle\text{número iniciado}\rangle \\ \langle\text{dígito no cero}\rangle & \to & 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\ \langle\text{dígito}\rangle & \to & 0 \mid \langle\text{dígito no cero}\rangle \end{array} \right\}$$

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:

$$V' = \{S, X, Y, Z\}$$
$$T' = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$
$$S' = S$$
$$P' = \left\{ \begin{array}{rcl} S & \to & 0 \mid 1N \mid 2N \mid 3N \mid 4N \mid 5N \mid 6N \mid 7N \mid 8N \mid 9N \\ N & \to & 0N \mid 1N \mid 2N \mid 3N \mid 4N \mid 5N \mid 6N \mid 7N \mid 8N \mid 9N \mid \varepsilon \end{array} \right\}$$

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

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

$$V = \{S, X, Y\}$$
$$T = \{a, b\}$$
$$S = S$$
$$P = \left\{ \begin{array}{rcl} S & \to & X \mid Y \mid \varepsilon \\ X & \to & aX \mid \varepsilon \\ Y & \to & aYb \mid \varepsilon \end{array} \right\}$$

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:

$$V = \{S, X, Y, Z\}$$
$$T = \{a, b, c\}$$
$$S = S$$
$$P = \left\{ \begin{array}{rcl} S & \to & aXbbcY \\ X & \to & aXbb \mid \varepsilon \\ Y & \to & cY \mid \varepsilon \end{array} \right\}$$

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

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

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

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:

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

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

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

   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:

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

Notemos que las reglas $S \to a \mid b$ 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. $\{uv \in \{0,1\}^* \mid u^{-1}$ es un prefijo de $v\}$

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

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

   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:

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

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

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

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

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

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

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

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:

$$
\begin{aligned}
V &= \{S, X\} \\
T &= \{a, b, c\} \\
S &= S \\
P &= \left\{ \begin{array}{lcl} S & \to & aSc \mid X \\ X & \to & bXc \mid \varepsilon \end{array} \right\}
\end{aligned}
$$

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:

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

   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:

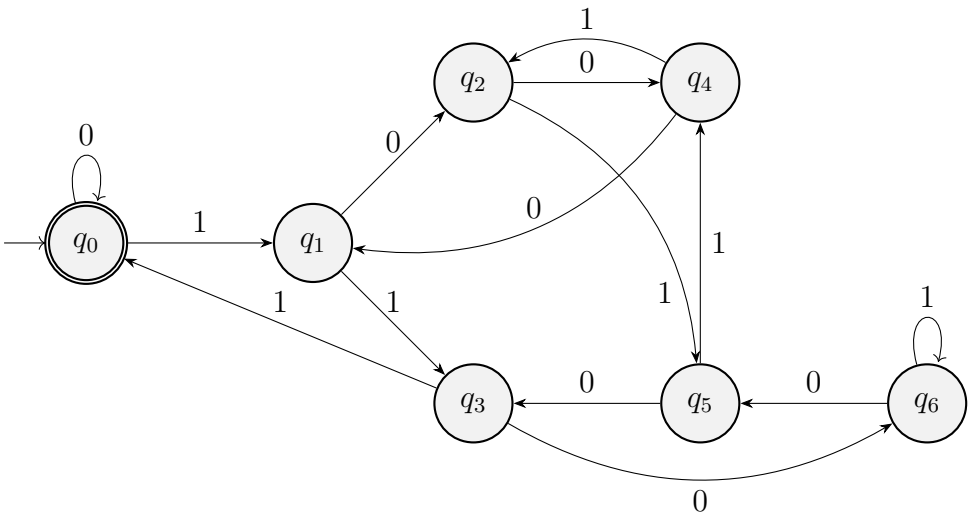


Figura 1.1: AFD que acepta los múltiplos de 7 en binario.

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

dada por:

$$
\begin{aligned}
&V = Q = \{q_0, \ldots, q_6\} \\
&T = A = \{0, 1\} \\
&S = q_0 \\
&P = \begin{cases}
q_0 & \to & 0q_0 \mid 1q_1 \mid \varepsilon \\
q_1 & \to & 0q_2 \mid 1q_3 \\
q_2 & \to & 0q_4 \mid 1q_5 \\
q_3 & \to & 0q_6 \mid 1q_0 \\
q_4 & \to & 0q_1 \mid 1q_2 \\
q_5 & \to & 0q_3 \mid 1q_4 \\
q_6 & \to & 0q_5 \mid 1q_6
\end{cases}
\end{aligned}
$$

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) Descarar el bit de acarreo (el más significativo).

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

$$
\begin{aligned}
&V = \{S, N, \alpha, \beta, \delta, \gamma, Z, Z', A, D, E, E_0, E_1, E_2, E_0', E_1', \overline{E_0}, \overline{E_1}, \overline{E_2}, L_0, L_1, X\} \\
&T = \{0, 1\} \\
&S = S
\end{aligned}
$$

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:

  $$S \to \alpha BNABBB\gamma \mid 0$$

  Iremos usando N para generar nuestro número a su izquierda y el complemento a 1 en espejo a la derecha.
  - Las tres Bs 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:

  $$N \to ANB \mid BNA \mid AAA\gamma\beta Z$$

  Una vez generado el número, terminaremos añadiendo 3 As 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:

  $$ZB \to BZ$$

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

  $$ZA \to Z'B$$

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

  $$BZ' \to Z'A$$

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

  $$\beta Z' \to \beta L_0$$

  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. Distinguimos dos casos:

  a) No nos llevamos nada de la operación anterior (L_0):

  $$L_0 A \to E_0$$

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

  $$AE_0 \to E_0 A$$
  $$BE_0 \to E_0 B$$
  $$\beta E_0 \to E_0 \beta$$

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

  $$\delta E_0 \to \overline{E_0}$$

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

  $$A\overline{E_0} \to \delta A E_0'$$
  $$B\overline{E_0} \to \delta B E_0'$$

  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:

  $$E_0' A \to A E_0'$$
  $$E_0' B \to B E_0'$$
  $$E_0' \beta \to \beta L_0$$

  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:

  $$L_0 B \to E_1$$
  $$AE_1 \to E_1 A$$
  $$BE_1 \to E_1 B$$
  $$\beta E_1 \to E_1 \beta$$
  $$\delta E_1 \to \overline{E_1}$$

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

  $$A\overline{E_1} \to \delta B E_0'$$
  $$B\overline{E_1} \to \delta A E_1'$$

- 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:

$$E_1'A \to AE_1'$$
$$E_1'B \to BE_1'$$
$$E_1'\beta \to \beta L_1$$

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:

$$L_1A \to E_1$$

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:

$$L_1B \to E_2$$

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):

$$AE_2 \to E_2A$$
$$BE_2 \to E_2B$$
$$\beta E_2 \to E_2\beta$$
$$\delta E_2 \to \overline{E_2}$$

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

$$A\overline{E_2} \to \delta AE_1'$$
$$B\overline{E_2} \to \delta BE_1'$$

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

- 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:

$$\beta L_1\gamma \to X$$

donde X es una variable finalizadora, que usamos para cambiar las As por 0s, las Bs por 1s 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):

$$AX \to X0$$
$$BX \to X1$$
$$\delta X \to X$$
$$\alpha X \to \varepsilon$$

Cuando lleguemos a αX, habremos "limpiado" la palaba, 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 &\to \alpha BNABBB\gamma \to \alpha BANBABBB\gamma \to \alpha BAAAA\delta\beta ZBABBB\gamma \to \\
&\to \alpha BAAAA\delta\beta BZABBB\gamma \to \alpha BAAAA\delta\beta BZ'BBBB\gamma \to \\
&\to \alpha BAAAA\delta\beta Z'ABBBB\gamma \to \alpha BAAAA\delta\beta L_0ABBBB\gamma \to \\
&\to \alpha BAAAA\delta\beta E_0BBBB\gamma \to \alpha BAAAA\delta E_0\beta BBBB\gamma \to \\
&\to \alpha BAAAA\overline{E_0}\beta BBBB\gamma \to \alpha BAAA\delta AE_0'\beta BBBB\gamma \to \\
&\to \alpha BAAA\delta A\beta L_0BBBB\gamma \to \alpha BAAA\delta A\beta E_1BBB\gamma \to \\
&\to \alpha BAAA\delta AE_1\beta BBB\gamma \to \alpha BAAA\delta E_1A\beta BBB\gamma \to \\
&\to \alpha BAAA\overline{E_1}A\beta BBB\gamma \to \alpha BAA\delta BE_0'A\beta BBB\gamma \to \\
&\to \alpha BAA\delta BAE_0'\beta BBB\gamma \to \alpha BAA\delta BA\beta L_0BBB\gamma \to \\
&\to \alpha BAA\delta BA\beta E_1BB\gamma \to \alpha BAA\delta BAE_1\beta BB\gamma \to \\
&\to \alpha BAA\delta BE_1A\beta BB\gamma \to \alpha BAA\delta E_1BA\beta BB\gamma \to \\
&\to \alpha BAA\overline{E_1}BA\beta BB\gamma \to \alpha BA\delta BE_0'BA\beta BB\gamma \to \\
&\to \alpha BA\delta BBE_0'A\beta BB\gamma \to \alpha BA\delta BBAE_0'\beta BB\gamma \to \\
&\to \alpha BA\delta BBA\beta L_0BB\gamma \to \alpha BA\delta BBA\beta E_1B\gamma \to \alpha BA\delta BBAE_1\beta B\gamma \to \\
&\to \alpha BA\delta BBE_1A\beta B\gamma \to \alpha BA\delta BE_1BA\beta B\gamma \to \alpha BA\delta E_1BBA\beta B\gamma \to \\
&\to \alpha BA\overline{E_1}BBA\beta B\gamma \to \alpha B\delta BE_0'BBA\beta B\gamma \to \alpha B\delta BBE_0'BA\beta B\gamma \to \\
&\to \alpha B\delta BBBE_0'A\beta B\gamma \to \alpha B\delta BBBAE_0'\beta B\gamma \to \alpha B\delta BBBA\beta L_0B\gamma \to \\
&\to \alpha B\delta BBBA\beta E_1\gamma \to \alpha B\delta BBBAE_1\beta\gamma \to \alpha B\delta BBBE_1A\beta\gamma \to \\
&\to \alpha B\delta BBE_1BA\beta\gamma \to \alpha B\delta BE_1BBA\beta\gamma \to \alpha B\delta E_1BBBA\delta\gamma \to \\
&\to \alpha B\overline{E_1}BBBA\beta\gamma \to \alpha\delta AE_1'BBBA\beta\gamma \to \alpha\delta ABE_1'BBA\beta\gamma \to \\
&\to \alpha\delta ABBE_1'BA\beta\gamma \to \alpha\delta ABBBE_1'A\beta\gamma \to \alpha\delta ABBBAE_1'\beta\gamma \to \\
&\to \alpha\delta ABBBA\beta L_1\gamma \to \alpha\delta ABBBAX \to \alpha\delta ABBBX0 \to \alpha\delta ABBX10 \to \\
&\to \alpha\delta ABX110 \to \alpha\delta AX1110 \to \alpha\delta X01110 \to \alpha X01110 \to 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 cualqueir 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:

$$S \to \alpha X \beta$$
$$X \to 0X \mid 1X \mid E\gamma$$

- 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 $xE\gamma$ 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:

$$1E\gamma \to \gamma 1 E_1$$

  Posteriormente, movemos dicha variable a la derecha:

$$E_1 1 \to 1 E_1$$
$$E_1 0 \to 0 E_1$$

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

$$E_1 \beta \to E' \beta 1$$

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

$$0E\gamma \to \gamma 0 E_0$$

$$E_0 1 \to 1 E_0$$
$$E_0 0 \to 0 E_0$$

$$E_0 \beta \to E' \beta 0$$

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

$$1E' \to E'1$$
$$0E' \to E'0$$
$$\gamma E' \to E\gamma$$

- 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:

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

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, $(1011)^2$):

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

2. $\{a^{n^2} \in \{a\}^* \mid n \geqslant 0\}$
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):

$$\varepsilon$$
$$a$$
$$aaaa$$
$$aaaaaaaaa$$
$$aaaaaaaaaaaaaaaa$$

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

$$a$$
$$aa\ aa$$
$$aaa\ aaa\ aaa$$
$$aaaa\ aaaa\ aaaa\ aaaa$$

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:

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

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):

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

A continuación, usamos X para generar las As:

$$X \to AX \mid E$$

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:

$$AE\beta \to \beta E$$

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:

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

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 encuentre 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:

$$E\delta \to E_\sigma\delta$$
$$aE_\sigma \to E_\sigma a$$
$$IE_\sigma \to I\sigma R$$

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.

$$Ra \to aR$$

Nos movemos a la derecha

$$aR\delta \to L\delta a$$

Cuando lleguemos a δ, guardamos una "a" más como copiada.

$$aL \to La$$

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

$$\sigma L \to a\sigma R$$

Pegaremos una a tras σ y repetiremos el proceso.

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

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

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 $\overline{E}$:

$$\overline{E}a \to a\overline{E}$$

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

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

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

$$aE' \to E'a$$
$$IE' \to E'I$$
$$\beta E' \to E\beta$$

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

$$\alpha E \to Z$$

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

$$Z\beta \to Z$$
$$ZI \to Z$$
$$Za \to aZ$$
$$Z\delta \to Z$$
$$Z\gamma \to \varepsilon$$

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

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

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

Se subirá próximamente una gramática.

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

   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 As (variables) y de bs. Seguidamente, generaremos tantas Bs 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 Bs y no hemos sustituido todas las As, entonces era porque teníamos más as de las permitidas en este lenguaje ($n > m$), con lo que no podremos generar ninguna palabra. Si por el contrario sustituimos todas las As y nos siguen quedando Bs, eliminaremos todas las Bs 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 As, 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 Bs como número de bs 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:

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

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

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

   de forma que el número de Bs coincide con el cuadrado del número de bs.

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

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

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

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:

$$S \to \lambda G \mid \varepsilon$$
$$G \to AG$$

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:

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

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

$$X \to bX \mid E$$

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):

$$bE\varphi \to \varphi bE$$

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:

$$Eb \to bE$$
$$E\beta \to \beta E$$
$$EI \to IBE \qquad \text{(Añade una } B\text{)}$$
$$EB \to BE$$
$$E\delta \to E_\sigma\delta$$

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:

$$BE_\sigma \to E_\sigma B$$
$$IE_\sigma \to I\sigma R$$

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

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

e) El proceso terinará cuando no haya más Bs 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 Bs recién creado y usamos $\overline{E}$ para devolver δ al final del último grupo de Bs:

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

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

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

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:

$$\lambda A \dots Ab\alpha E\varphi b \dots b\beta IB \dots BIB \dots BI \dots IB \dots B\delta\gamma$$

donde tenemos tantas Bs como número de bs al cuadrado (gracias al funcionamiento de la gramáica 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:

$$\begin{aligned} \alpha E\varphi &\to Z \qquad \text{(No quedan } b\text{s por copiar)} \\ Zb &\to bZ \\ Z\beta &\to \beta Z \\ ZI &\to Z \\ ZB &\to BZ \\ Z\delta &\to H \end{aligned}$$

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

a) En primer lugar, borramos una B:

$$BBH \to H'B \qquad \text{(No es la última)}$$

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

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

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:

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

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

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

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

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

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 Bs que As. 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:

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

  Y la variable ψ deberá ser la encargada de eliminar las Bs 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 $\lambda A\overline{H} \to a\psi$, 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:

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

  que será imposible sustituir por un símbolo terminal (no tenemos ninguan 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:

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

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

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

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:

$$P' = \left\{ \begin{array}{ll} S & \to aS \mid B \\ B & \to b \end{array} \right\}$$

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:

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

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 $f : \mathbb{N} \to L^*$ 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 $L = \{u_1, u_2, \ldots, u_n\}$, podemos construir una gramática generativa de tipo 3 que nos genere dicho lenguaje (suponiendo que el lenguaje contiene palabras de un cierto alfabeto A):

   $$G = (\{S\}, A, P, S)$$
   $$P = \{ \ S \to u_1 \mid u_2 \mid \ \ldots \ \mid u_n \ \}$$

   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 afabeto, 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:

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

   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:

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

   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:

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

   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 leido 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:
    $$f(uv) = f(u)f(v) \qquad \forall u, v \in A^*$$
    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 $n \geqslant 2$ (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: $G = (\{S\}, \{a\}, \{S \to aS, S \to a\}, S)$ 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:

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

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

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

    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:

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

    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

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

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 $f : A^* \to A^*$ la aplicación enunciada, definida por:

    $$f(u) = a_1a_1a_2a_2 \dots a_na_n \qquad \forall u = a_1a_2 \dots a_n \quad a_i \in A \quad \forall i \in \{1, \dots, n\}$$

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

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

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

    $$\begin{aligned} u &= a_1a_2 \dots a_n \qquad a_i \in A \quad \forall i \in \{1, \dots, n\} \\ v &= b_1b_2 \dots b_m \qquad b_i \in A \quad \forall i \in \{1, \dots, m\} \end{aligned}$$

    De esta forma:

    $$\begin{aligned} f(uv) = f(a_1a_2 \dots a_nb_1b_2 \dots b_m) &= a_1a_2a_2a_2 \dots a_na_nb_1b_1b_2b_2 \dots b_mb_m = \\ &= f(a_1a_2 \dots a_n)f(b_1b_2 \dots b_m) = f(u)f(v) \end{aligned}$$

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

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

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

    $$u = a_1a_2 \dots a_n \qquad a_i \in A \quad \forall i \in \{1, \dots n\}$$

    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 afabeto, 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 $f : A^* \to A^*$ 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:

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

    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.

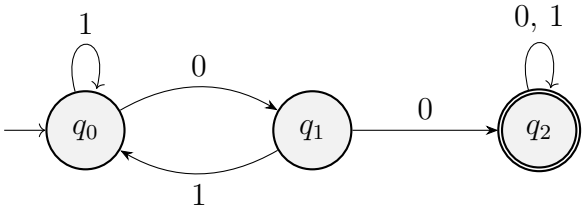


Figura 1.2: Autómata Finito Determinista del Ejercicio 1.2.1.

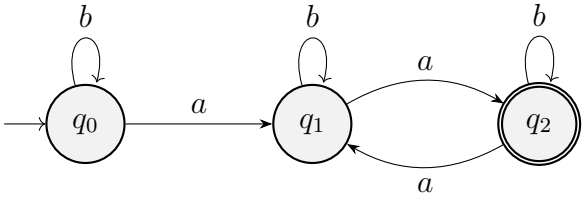


Figura 1.3: Autómata Finito Determinista del Ejercicio 1.2.2.

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:

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

- $F = \{q_2\}$

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,

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

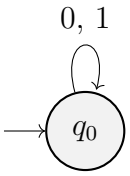
Ejercicio 1.2.2. Dado el AFD de la Figura 1.3, describir el lenguaje aceptado por dicho autómata.

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

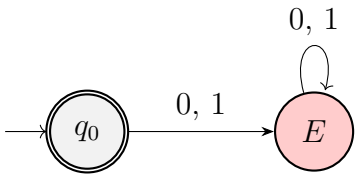
$$L = \{u \in \{a, b\}^* \mid n_a(u) \text{ es par}, \ n_a(u) > 0\},$$

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

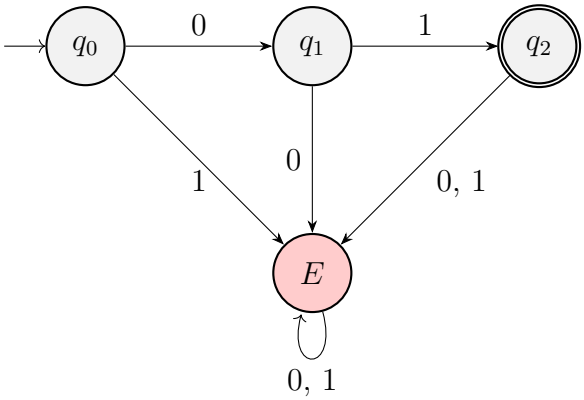
1. El lenguaje vacío,



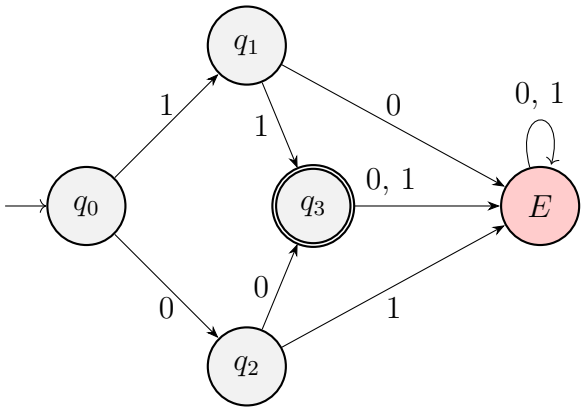
2. El lenguaje formado por la palabra vacía, es decir, $\{\varepsilon\}$,



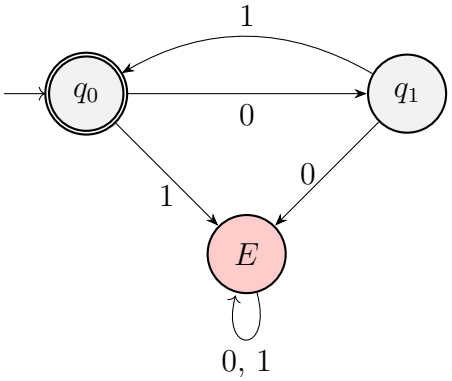
3. El lenguaje formado por la palabra 01, es decir, $\{01\}$,



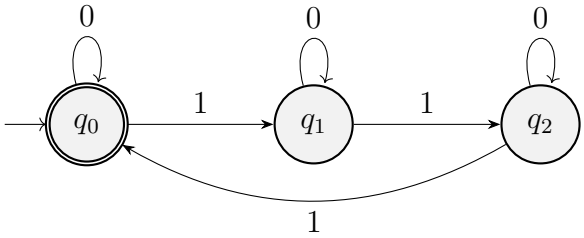
4. El lenguaje $\{11, 00\}$,



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



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

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

un AFND que reconozca el lenguaje generado por esa gramática.

El autómata obtenido es el de la Figura 1.4.

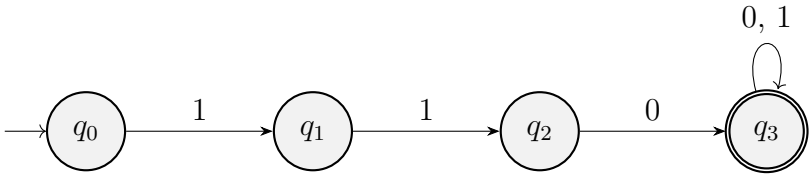


Figura 1.4: Autómata Finito No Determinista del Ejercicio 1.2.4.

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

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

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.

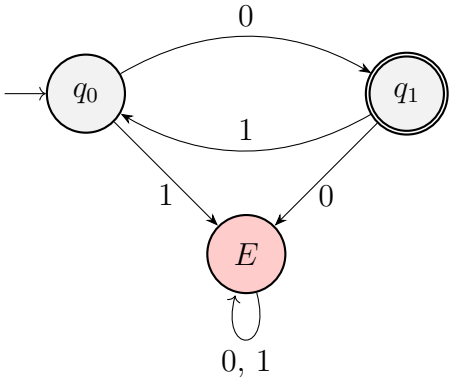


Figura 1.5: Autómata Finito Determinista del Ejercicio 1.2.5.

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.

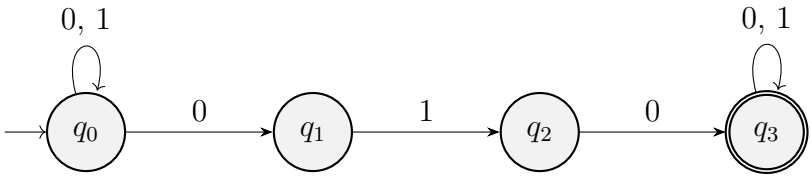


Figura 1.6: Autómata Finito No Determinista del Ejercicio 1.2.6 apartado 1.

2. AFND. Contengan la subcadena 110.

   El autómata obtenido es el de la Figura 1.7.

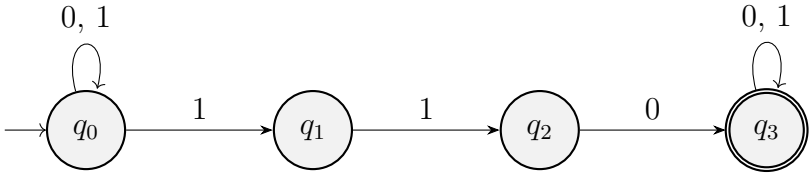


Figura 1.7: Autómata Finito No Determinista del Ejercicio 1.2.6 apartado 2.

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.

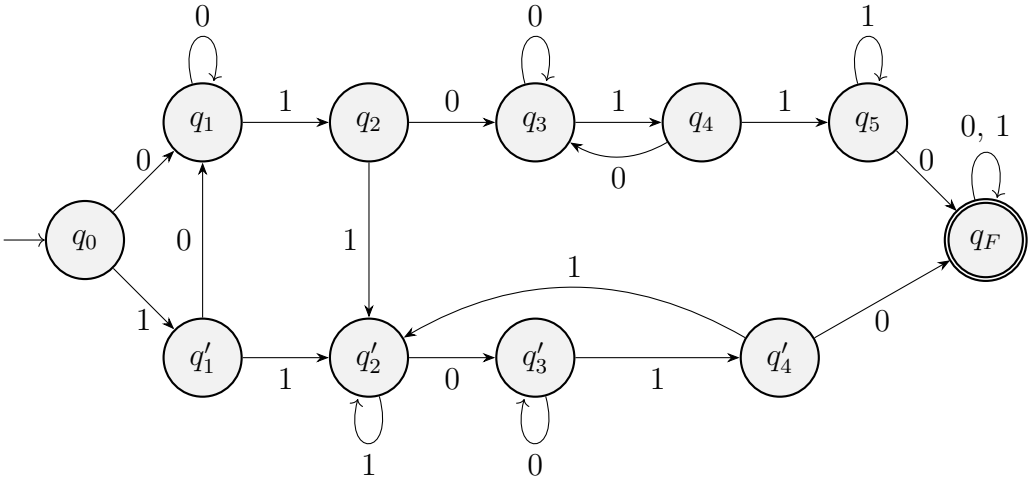


Figura 1.8: Autómata Finito Determinista del Ejercicio 1.2.6 apartado 3.

- 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:

$$S \to AB, \qquad A \to aA, \qquad A \to c, \qquad B \to bBb, \qquad B \to b.$$

El lenguaje generado por la gramática es:

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

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.

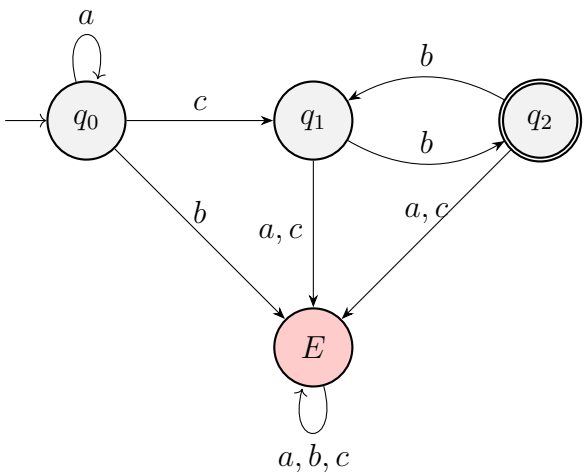


Figura 1.9: Autómata Finito Determinista del Ejercicio 1.2.7.

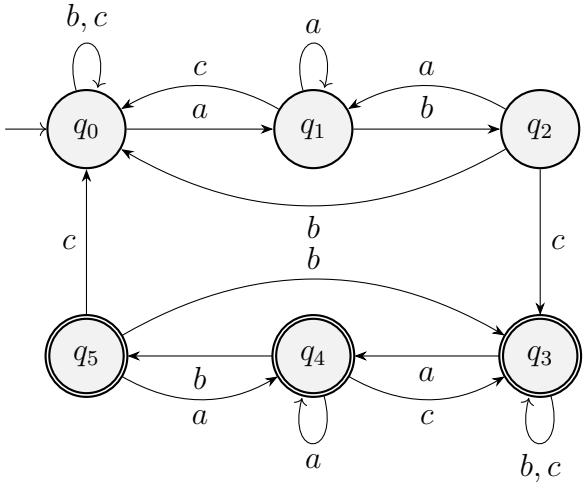


Figura 1.10: Autómata Finito Determinista del Ejercicio 1.2.8.

- 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

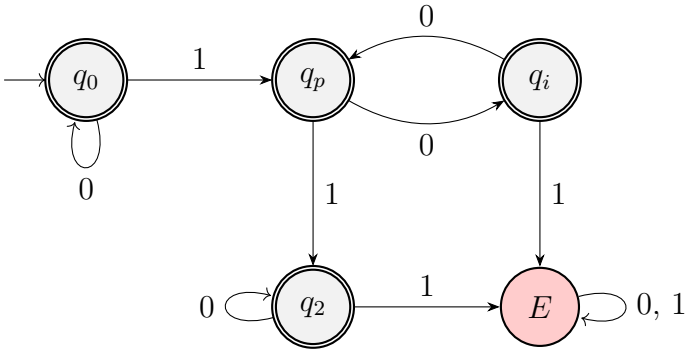


Figura 1.11: Autómata Finito Determinista del Ejercicio 1.2.9.

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

$$S \to abA \mid B \mid baB \mid \varepsilon, \qquad A \to bS \mid b, \qquad B \to aS.$$

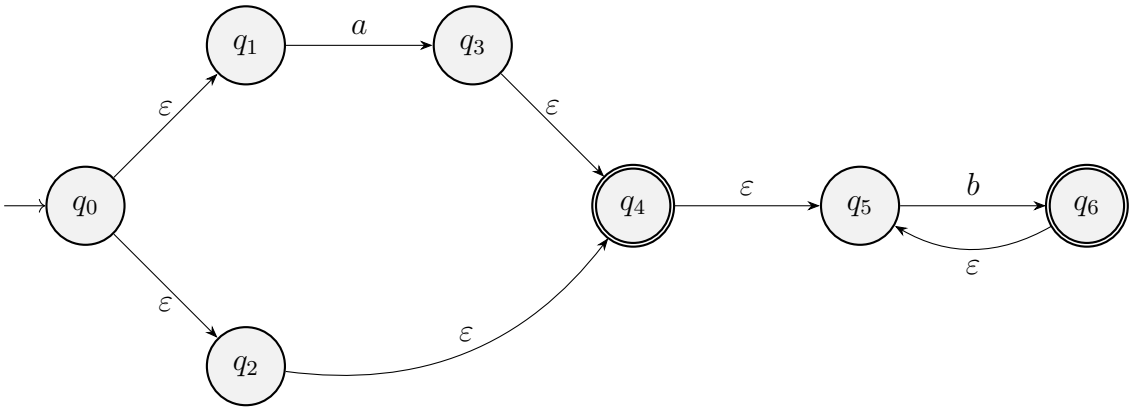


Figura 1.12: Autómata Finito No Determinista algorítmico del Ejercicio 1.2.10.

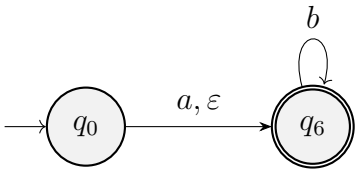


Figura 1.13: Autómata Finito No Determinista simplificado del Ejercicio 1.2.10.

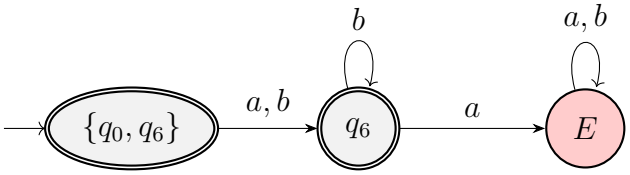


Figura 1.14: Autómata Finito Determinista del Ejercicio 1.2.10.

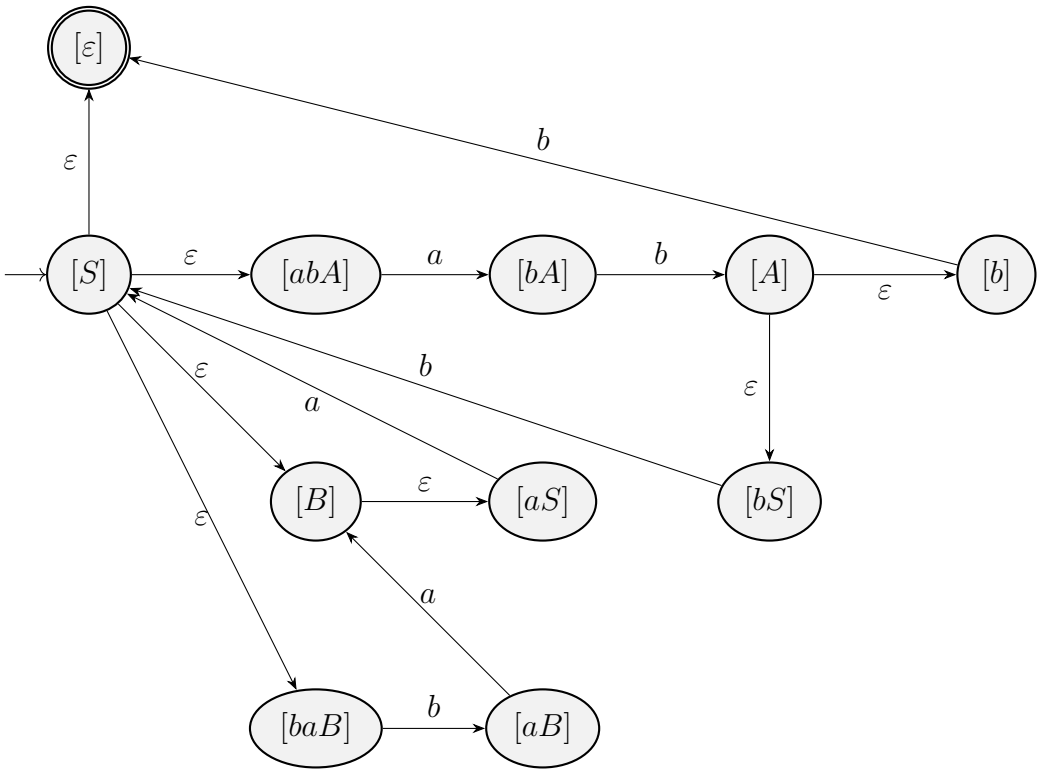


Figura 1.15: Autómata Finito Determinista del lenguaje $\mathcal{L}(S)$ del Ejercicio 1.2.11.

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:

$$S \to abA \mid aS \mid baaS \mid \varepsilon, \qquad A \to bS \mid b$$

Eliminamos ahora la variable A:

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

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

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

Por tanto, vemos que:

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

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

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

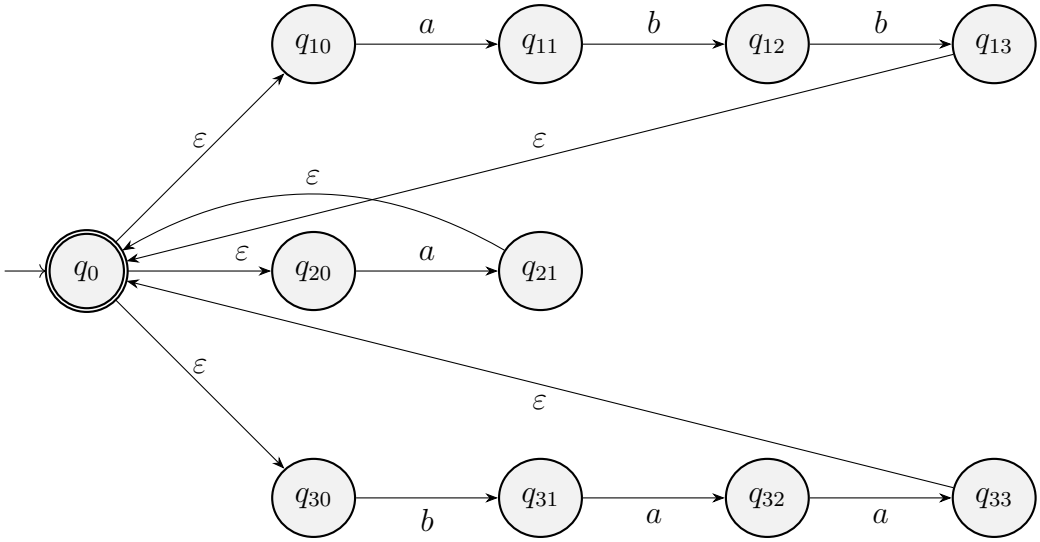


Figura 1.16: AFND asociado a $\mathcal{L}(G)$ del Ejercicio 1.2.11.

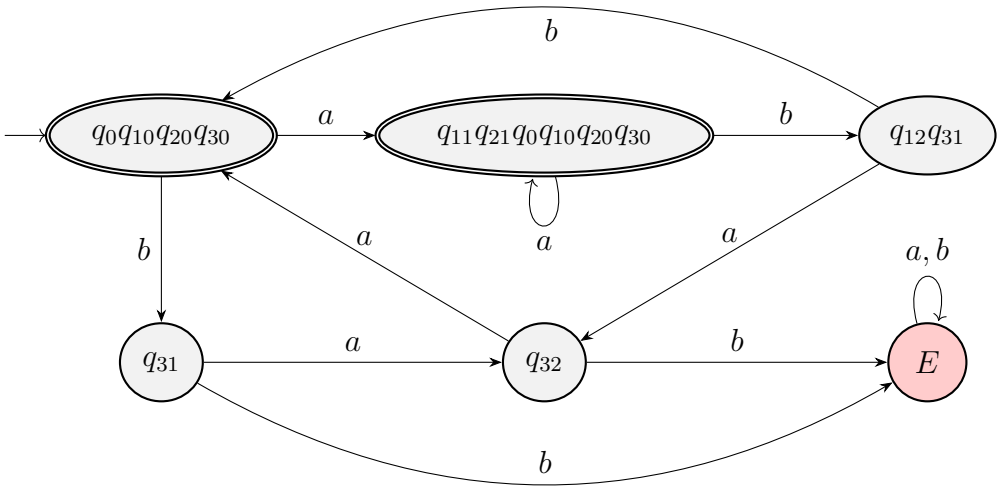


Figura 1.17: AFD asociado a $\mathcal{L}(G)$ del Ejercicio 1.2.11.

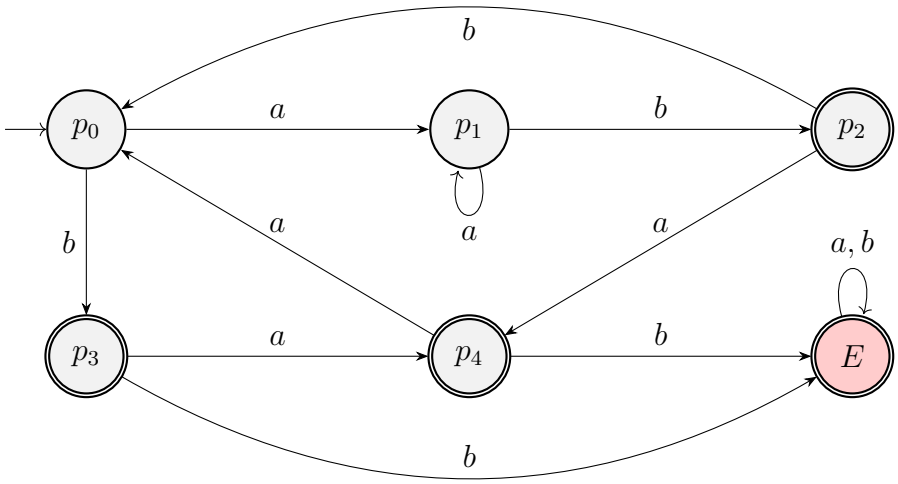


Figura 1.18: Autómata Finito Determinista del lenguaje $\overline{\mathcal{L}(G)}$ del Ejercicio 1.2.11.

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:

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

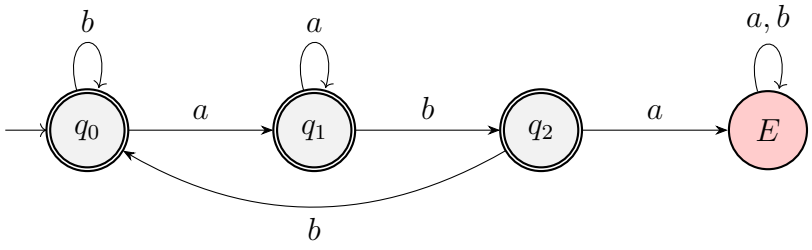


Figura 1.19: AFD del lenguaje del Ejercicio 1.2.12 apartado 3.

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:

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

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

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

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:

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

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:

   - $\underline{q_0}$: No se ha empezado ninguna subcadena.

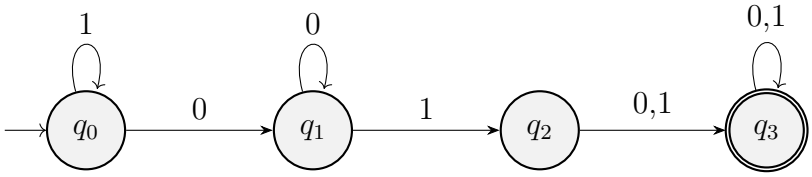


Figura 1.20: Autómata Finito Determinista del Ejercicio 1.2.14 apartado 1.

- $\underline{q_1}$: 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).

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

   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:

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

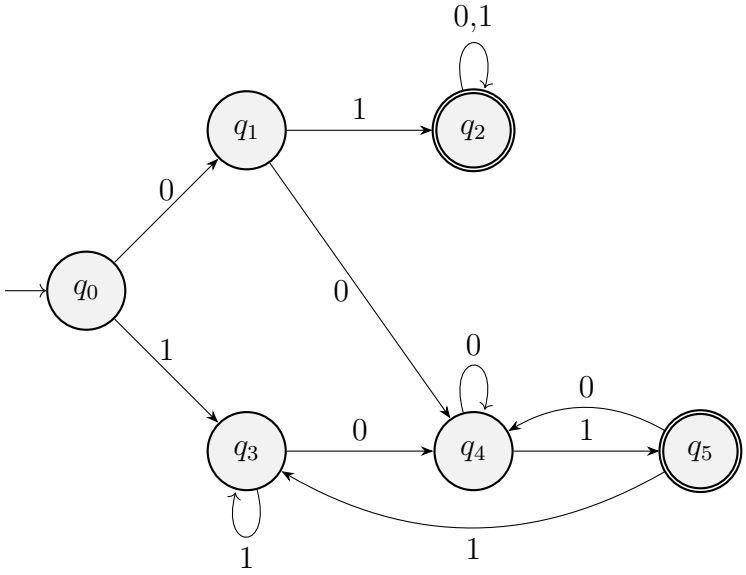


Figura 1.21: Autómata Finito Determinista del Ejercicio 1.2.14 apartado 2.

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:

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

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

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

La expresión regular es:

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

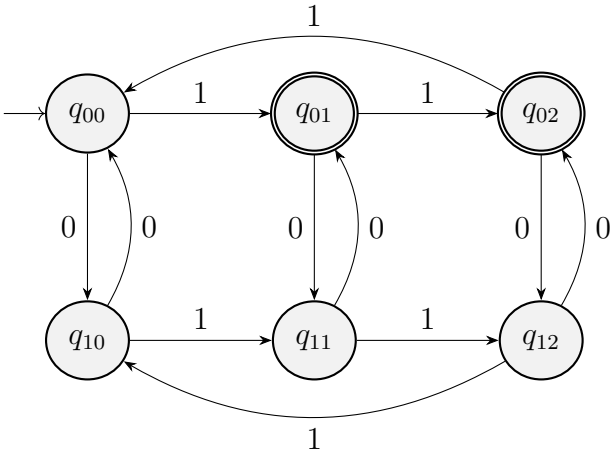


Figura 1.22: Autómata Finito Determinista del Ejercicio 1.2.17.

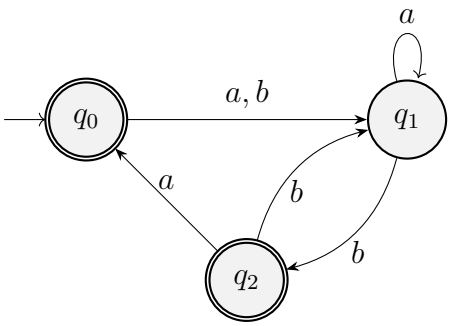


Figura 1.23: Autómata Finito Determinista del Ejercicio 1.2.18.

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

$L_3 = \{u \in \{0,1\}^* \mid$ 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:

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

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

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

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:

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

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

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

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

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

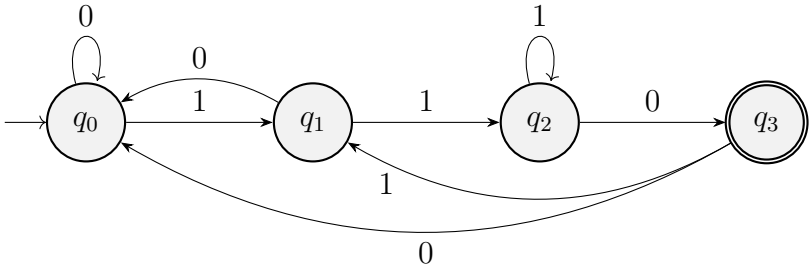


Figura 1.24: Autómata Finito Determinista del Ejercicio 1.2.19.

- $\underline{q_1}$: He leído un 1 de la cadena final.
- $\underline{q_2}$: He leído un 11 de la cadena final.
- $\underline{q_3}$: 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 $Q = \{q_1, q_2, \ldots, q_n\}$. 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:

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

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:

- $\underline{q_0}$: No se ha empezado la subcadena 001
- $\underline{q_1}$: Se ha leído un 0 de la subcadena 001.
- $\underline{q_2}$: Se ha leído un 00 de la subcadena 001.
- $\underline{E}$: Se ha leído la subcadena 001, por lo que es el estado de error.

El autómata obtenido es el de la Figura 1.25.

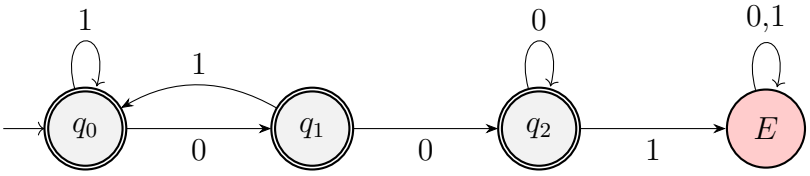


Figura 1.25: Autómata Finito Determinista del Ejercicio 1.2.21.

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:

$$P = \begin{cases} S & \to & q_0 \mid q_1 \mid q_2, \\ q_0 & \to & q_0 1 \mid q_1 1 \mid \varepsilon \\ q_1 & \to & q_0 0, \\ q_2 & \to & q_1 0 \mid q_2 0, \\ E & \to & E0 \mid E1 \mid q_2 1, \end{cases}$$

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:

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

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 $u \neq \varepsilon$. Demostrar que si L es regular, también lo son los lenguajes siguientes:

1. NOPREFIJO$(L) = \{u \in L \mid$ ningún prefijo propio de u 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 NOPREFIJO$(L) = \mathcal{L}(M')$.

   $\subseteq$) Sea $u \in$ NOPREFIJO(L). Entonces, por definición de 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 $L = \{a^i b^i \mid i \geqslant 0\}$.

   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:

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

   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 $j \leqslant i$, 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 $L = \{a^i b^j \mid i, j \geqslant 0\}$.

   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.

   $\Longrightarrow$) 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 indistingibles, 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:

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

Es decir:

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

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.

$\Longleftarrow$) 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 $u \in L \Longleftrightarrow v \in L$. Esto es directo tomando $z = \varepsilon$.

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

$$u \in \mathcal{L}(M) \Longleftrightarrow \delta^*(q_0, u) \in F \Longleftrightarrow \delta^*([\varepsilon], u) \in F \Longleftrightarrow [\varepsilon u] \in F \Longleftrightarrow [u] \in F \Longleftrightarrow$$
$$\Longleftrightarrow u \in L$$

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 indistingibles.

- 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 $\mathrm{Per}(u)$ al conjunto

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

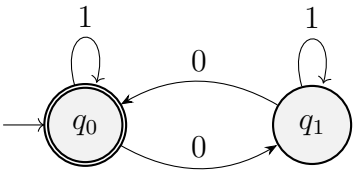
Dado un lenguaje L, se llama $\mathrm{Per}(L) = \bigcup_{u \in L} \mathrm{Per}(u)$. Dar expresiones regulares y autómatas minimales para $\mathrm{Per}(L)$ en los siguientes casos:

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

   Tenemos que:

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

   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:

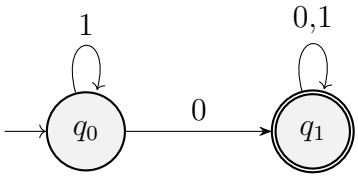
$$q_0 = 1q_0 + 0(1^*0q_0) + \varepsilon$$
$$q_0 = 1q_0 + 01^*0q_0 + \varepsilon$$
$$q_0 = (1 + 01^*0)q_0 + \varepsilon$$
$$q_0 = (1 + 01^*0)^*$$

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 $z = 0^n1^n$, que cumple $|z| \geqslant n$. Si consideramos una descomposición $z = uvw$ con $|uv| \leqslant n$ y $|v| \geqslant 1$, tenemos que:

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

   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 $L_2 = \{0^i 1^j \mid i, j \in \mathbb{N}\}$.

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

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

     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:

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

      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:

      $$u = v_1 v_2 \dots v_n \qquad v_i \in R \quad \forall i \in \{1, \dots, n\} \qquad n > 1$$

      por lo que podemos tomar $w = v_2 \dots v_n \in R^*$ y $v_1 \in R \subseteq R^*$, llegando a que

      $$u = v_1 w \qquad v_1, w \in R^* \Longrightarrow u \in R^* R^*$$

- 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

      $$u = v_0w_1v_1w_2v_2 \dots w_nv_n \qquad v_i \in R_1 \quad w_i \in R_2$$

      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

    $$u = v_1v_1'v_2v_2' \dots v_nv_n' \qquad v_i, v_i' \in R \quad \forall i \in \{1, \dots, n\}$$

    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 determistas, 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 $L = \{u_1, u_2, \ldots, u_n\}$ 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 $i \in \{1, \ldots, n\}$ de longitud $k = |u_i|$, añadimos k nuevos estados a los que llamaremos por ejemplo q_{ij} con $j \in \{1, \ldots, k\}$. De esta forma, si

$$u_i = a_{i1}a_{i2}\ldots a_{ik} \qquad a_{ij} \in A \quad \forall j \in \{1, \ldots, k\}$$

Entonces, añadimos las transiciones $\delta(q_{ij-1}, a_j) = q_{ij}$, entendiendo que $q_{i0} = q_0$ para cualquier $i \in \{1, \ldots, n\}$ y consideraremos que q_{ik} es un estado final.

De manera más formal, dado un lenguaje finito $L = \{u_1, u_2, \ldots, u_n\}$ sobre un alfabeto A, construimos el autómata $M = (Q, A, \delta, q_0, F)$ formado por los conjuntos:

$$Q = \{q_0\} \cup \{q_{ij_i} \mid i \in \{1, \ldots, n\}, j_i \in \{1, \ldots, |u_i|\}\}$$
$$F = \{q_{ij_i} \mid j_i = |u_i|\}$$

donde la función de transición δ viene dada por:

$$\delta(q_0, a) = \begin{cases} \{q_{i1}\} & \text{si} \quad a = a_{i1} \\ \emptyset & \text{si} \quad a \neq a_{i1} \quad \forall i \in \{1, \ldots, n\} \end{cases}$$

$$\delta(q_{ij_i}, a) = \begin{cases} \{q_{ij_i+1}\} & \text{si} \quad a = a_{ij_i} \\ \emptyset & \text{si} \quad a \neq a_{ij_i} \quad \forall i \in \{1, \ldots, n\}, j_i \in \{1, \ldots, |u_i|\} \end{cases}$$

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 $L = \{u_1, u_2, \ldots, u_n\}$ es un lenguaje finito, entonces podemos considerar la gramática $G = (\{S\}, A, P, S)$ donde P es el conjunto:

$$P = \{S \to u_1, S \to u_2, \ldots, S \to u_n\}$$

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:

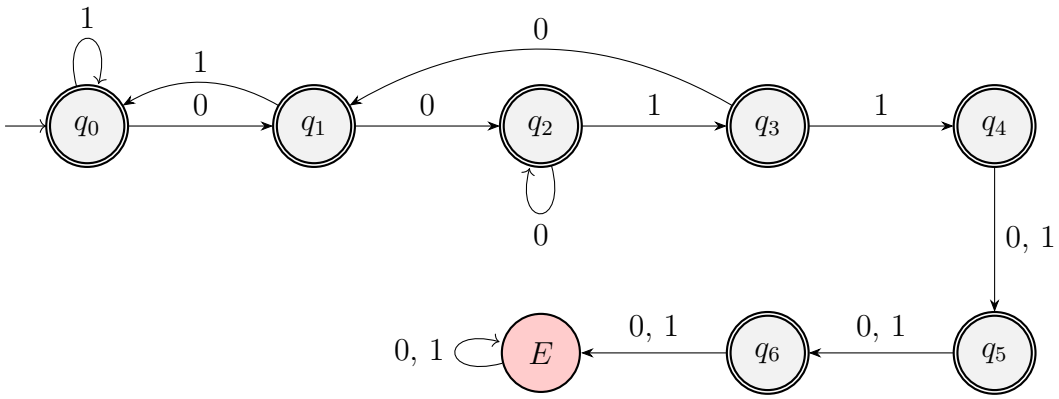


Figura 1.26: Autómata Finito Determinista para la pregunta 34.

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:

$$
\begin{aligned}
A &\to 0B \mid 1C \\
B &\to \varepsilon \mid 0A \mid 1D \\
C &\to 1A \mid 0D \\
D &\to 1B \mid 0C
\end{aligned}
$$

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 $|z| = 3n \geqslant n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0, b\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ debe cumplir que:

   $$u = 0^k \quad v = 0^l \quad w = 0^{2n-k-l}b^n \qquad \text{con } l, k \in \mathbb{N} \cup \{0\},\ l \geqslant 1,\ k + l \leqslant n$$

   Para $i = 2$, tenemos que $uv^iw = 0^{k+2l+2n-k-l}b^n = 0^{2n+l}b^n \notin L_1$, ya que, como $l \geqslant 1$:

   $$2n + l \neq 2n \quad \text{y} \quad 2(2n + l) \neq n$$

   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:

   $$\begin{aligned} S &\to S_1 \mid S_2 \\ S_1 &\to 00S_1b \mid \varepsilon \\ S_2 &\to 0S_2bb \mid \varepsilon \end{aligned}$$

   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. $L_2 = \{uu^{-1} \mid u \in \{0, 1\}^*, |u| \leqslant 1000\}$

   Consideramos el lenguaje auxiliar:

   $$L_2' = \{u \in \{0, 1\}^* \mid |u| \leqslant 2 \cdot 1000\}$$

   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. $L_3 = \{uu^{-1} \mid u \in \{0, 1\}^*, |u| \geqslant 1000\}$

   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:

$$V = \{S\} \cup \{A_i \mid i \in \{1, \ldots, 1000\}\}$$

Tenemos que P está definido por:

$$\begin{aligned} S &\to 0A_1 0 \mid 1A_1 1, \\ A_i &\to 0A_{i+1}0 \mid 1A_{i+1}, \qquad i \in \{1, \ldots, 999\}, \\ A_{1000} &\to 0A_{1000}0 \mid 1A_{1000}1 \mid \varepsilon \end{aligned}$$

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 $|z| = 3n \geqslant n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0,1,2\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ debe cumplir que:

   $$u = 0^k \quad v = 0^l \quad w = 0^{n-k-l}1^n 2^{2n} \qquad \text{con } l, k \in \mathbb{N} \cup \{0\},\ l \geqslant 1,\ k + l \leqslant n$$

   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 $l \geqslant 1$:

   $$n + l \neq n \quad \text{y} \quad n \neq 2n$$

   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:

   $$\begin{aligned} S &\to S_1 A_2 \mid A_0 S_2 \\ S_1 &\to 0S_1 1 \mid \varepsilon \\ A_2 &\to 2A_2 \mid \varepsilon \\ S_2 &\to 1S_2 2 \mid \varepsilon \\ A_0 &\to 0A_0 \mid \varepsilon \end{aligned}$$

   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 $|z| = 2n+1 \geqslant n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0,1\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ debe cumplir que:

   $$u = 0^k \quad v = 0^l \quad w = 0^{n-k-l}10^n \qquad \text{con } l, k \in \mathbb{N} \cup \{0\},\ l \geqslant 1,\ k + l \leqslant n$$

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 $l \geqslant 1$:

$$n + l \neq n$$

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:

$$S \to 0S01S1 \mid 0 \mid 1$$

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.

   Por tanto, como hemos dado un autómata que reconoce el lenguaje, es regular.

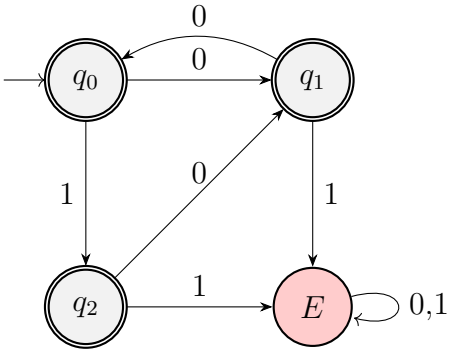


Figura 1.27: Autómata que reconoce el lenguaje del ejercicio 1.3.3.1.

2. Conjunto $\{0^i 1^{2j} 0^{i+j} \mid i, j \geqslant 0\}$

   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 $|z| = 5n \geqslant n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0,1\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ debe cumplir que:

   $$u = 0^k \quad v = 0^l \quad w = 0^{n-k-l} 1^{2n} 0^{2n} \qquad \text{con } l, k \in \mathbb{N} \cup \{0\},\ l \geqslant 1,\ k + l \leqslant n$$

   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 $l \geqslant 1$:

   $$2n \neq n + n + l$$

   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:

   $$S \to 11S0 \mid X,$$
   $$X \to 0X0 \mid \varepsilon.$$

   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 $\{0^i 1^j 0^{i*j} \mid i, j \geqslant 0\}$

   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 $|z| = n + n + n^2 \geqslant n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0,1\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ debe cumplir que:

   $$u = 0^k \quad v = 0^l \quad w = 0^{n-k-l} 1^n 0^{n^2} \qquad \text{con } l, k \in \mathbb{N} \cup \{0\},\ l \geqslant 1,\ k + l \leqslant n$$

   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 $l \geqslant 1$:

   $$(n + l) \cdot n = n^2 + nl \neq n^2$$

   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 $|z| = 2n+1 \geqslant n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0,1\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ debe cumplir que:

   $$u = 0^k \quad v = 0^l \quad w = 0^{n+1-k-l}1^n \qquad \text{con } l, k \in \mathbb{N} \cup \{0\},\ l \geqslant 1,\ k + l \leqslant n$$

   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:

   $$n < n + 1 - l \Longleftarrow l < 1$$

   Pero esto es una contradicción, ya que $l \geqslant 1$. 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:

   $$S \to 0S \mid 0S'$$
   $$S' \to 0S'1 \mid \varepsilon$$

   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 $S \to 0S'$ fuerza a que haya al menos un 0 más que 1, y la producción $S \to 0S$ permite que la diferencia no sea de una sola unidad, sino que pueda ser mayor.

2. $L_2 = \{001^i 0^j \mid i, j \geqslant 1\}$.

   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:

   $$(0+1)^*010(0+1)^*$$

   Por tanto, $\overline{L'}$ es regular, por lo que está asociado a una expresión regular, sea esta $\widetilde{r}$. Entonces, L_3 es regular y está asociado a la expresión regular:

   $$010\widetilde{r}$$

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 $|w| = n+3+n \geqslant n$. Toda descomposición $w = uvw$, con $u, v, w \in \{0, 1, +, =\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ debe cumplir que:

$$u = 1^k \quad v = 1^l \quad w = [1^{n-k-l} = 0 + 1^n] \qquad \text{con } l, k \in \mathbb{N} \cup \{0\},\ l \geqslant 1,\ k + l \leqslant n$$

Para $i = 2$, tenemos que $uv^iw = [1^{k+2l+n-k-l} = 0+1^n] = [1^{n+l} = 0+1^n] \notin \text{ADD}$, ya que, como $l \geqslant 1$, 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:

   $\subset$) Se tiene trivialmente que $L \subset \{0, 1\}^*$.

   $\supset$) 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 $z = 1^n01^n \in L$ con $|z| = 2n + 1 \geqslant n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0, 1\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ debe cumplir que:

   $$u = 1^k \quad v = 1^l \quad w = 1^{n-k-l}01^n \qquad \text{con } l, k \in \mathbb{N} \cup \{0\},\ l \geqslant 1,\ k + l \leqslant n$$

   Para $i = 2$, tenemos que $uv^iw = 1^{k+2l+n-k-l}01^n = 1^{n+l}01^n \notin L$, ya que, como $l \geqslant 1$:

   $$n + l \neq n \Longrightarrow (1^{n+l})^{-1} = 1^{n+l} \neq 1^n$$

   Por tanto, por el recíproco del Lema de Bombeo, no es regular.

3. L es el lenguaje sobre el alfebeto $\{0, 1\}$ formado por las palabres en las que el tercer símbolo empezando por el final es un 1.

   Este lenguaje es regular, con reconocedor:

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

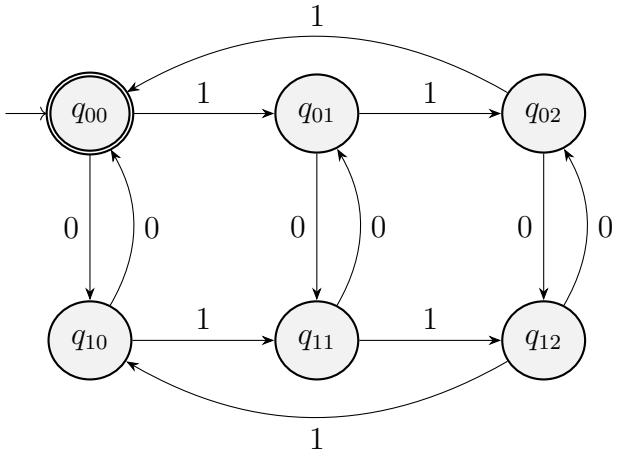


Figura 1.28: Autómata que reconoce el lenguaje del ejercicio 1.3.7.1.

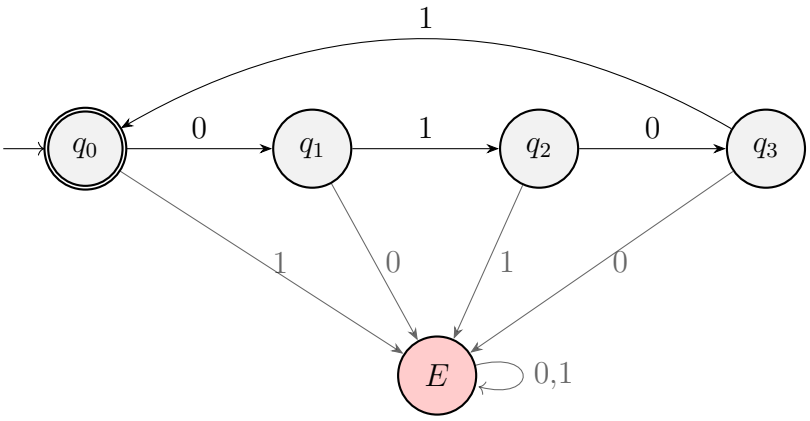


Figura 1.29: Autómata que reconoce el lenguaje del ejercicio 1.3.7.2.

Ejercicio 1.3.7. Obtener autómatas finitos determinísticos para los siguientes lenguajes sobre el alfabeto $\{0, 1\}$.

1. Palabras en las que el número de 1 es múltiplo de 3 y el número de 0 es par.

   El estado q_{ij} para $i = 0, 1, 2$ y $j = 0, 1$ indica que:

   - $n_1(u)\text{mod}3 = i$,
   - $n_0(u)\text{mod}2 = j$.

   Entonces, el autómata es el de la Figura 1.28.

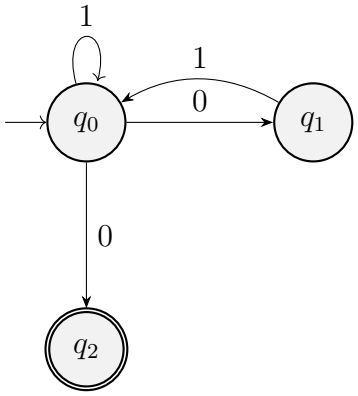
2. $\{(01)^{2i} \mid i \geqslant 0\}$

   El autómata es el de la Figura 1.29.

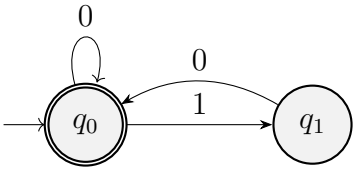
3. $L_3 = \{(0^{2i}1^{2i}) \mid i \geqslant 0\}$

   Veamos que este lenguaje no es regular con el Lema de Bombeo. Para todo $n \in \mathbb{N}$, consideramos la palabra $z = 0^{2n}1^{2n} \in L_3$ con $|z| = 4n \geqslant n$. Toda descomposición $z = uvw$, con $u, v, w \in \{0, 1\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ debe cumplir que:

   $$u = 0^k \quad v = 0^l \quad w = 0^{2n-k-l}1^{2n} \qquad \text{con } l, k \in \mathbb{N} \cup \{0\},\ l \geqslant 1,\ k + l \leqslant n$$

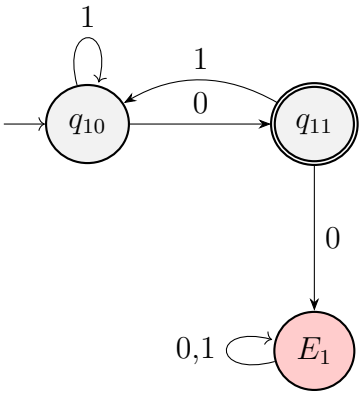


(a) AFND asociado a r_1.

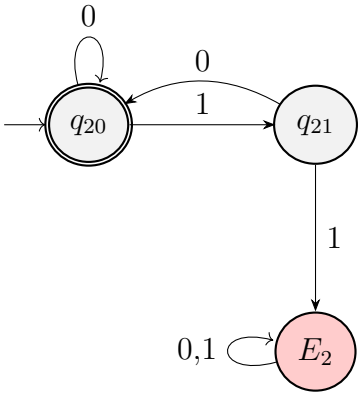


(b) AFND asociado a r_2.

Figura 1.30: AFND asociados a las expresiones regulares r_1 y r_2.



(a) AFD asociado a r_1.



(b) AFD asociado a r_2.

Figura 1.31: AFD asociados a las expresiones regulares r_1 y r_2.

Para $i = 2$, tenemos que $uv^iw = 0^{k+2l+2n-k-l}1^{2n} = 0^{2n+l}1^{2n} \notin L_3$, ya que, como $l \geqslant 1$:

$$2n + l \neq 2n$$

Por tanto, por el recíproco del Lema de Bombeo, no es regular. Por tanto, no es posible construir un autómata finito determinístico que reconozca L_3.

Ejercicio 1.3.8. 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 1.30a y 1.30b, respectivamente.

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 1.31a y 1.31b, respectivamente.

Por tanto, el AFD que acepta la intersección de los lenguajes asociados a r_1 y r_2 es el de la Figura 1.32.

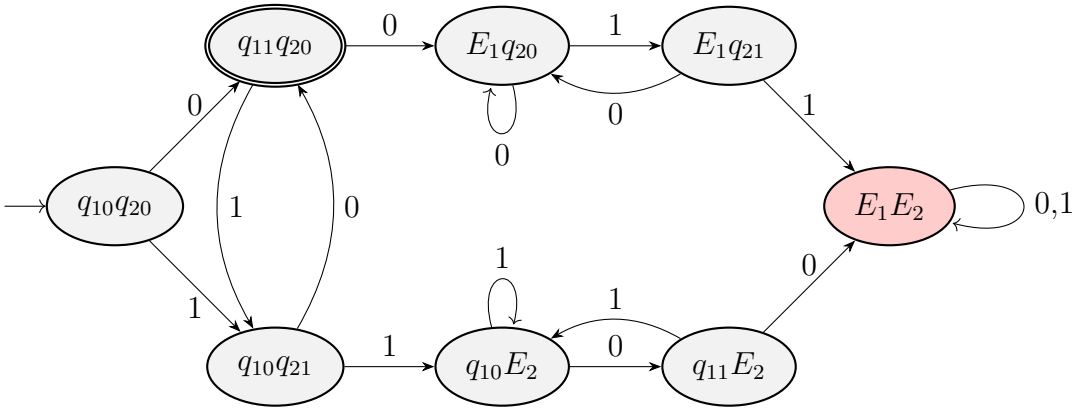


Figura 1.32: AFD que acepta la intersección de los lenguajes asociados a r_1 y r_2.

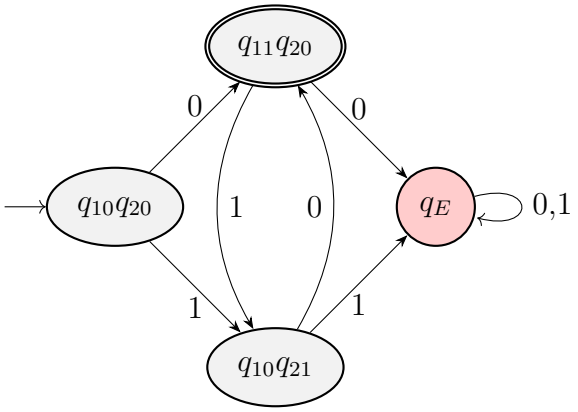


Figura 1.33: AFD minimal que acepta la intersección de los lenguajes asociados a r_1 y r_2.

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 1.33.

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:

$$\delta(q_E, 0) = q_E \notin F \qquad \delta(q_{10}q_{20}, 0) = \delta(q_{10}q_{21}, 0) = q_{11}q_{20} \in F$$

- $q_{10}q_{20}$ y $q_{10}q_{21}$ son distinguibles, ya que leyendo 10:

$$\delta(q_{10}q_{20}, 1) = q_{11}q_{20} \in F \qquad \delta(q_{10}q_{21}, 1) = q_E \notin F$$

Por tanto, el AFD minimal que acepta la intersección de los lenguajes asociados a r_1 y r_2 es el de la Figura 1.33. Para obtener la expresión regular asociada, resolvemos

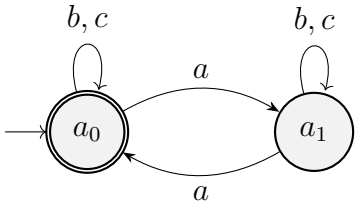


Figura 1.34: AFD que acepta palabras con número par de a's.

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:

$$q_{11}q_{20} = 1q_{11}q_{20} + \varepsilon \Longrightarrow q_{11}q_{20} = 10q_{11}q_{20} + \varepsilon = (10)^*\varepsilon = (10)^*$$

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 1.33 es

$$(0+10)(10)^*.$$

Ejercicio 1.3.9. Construir un Autómata Finito Determinista Minimal que acepte el lenguaje sobre el alfabeto $\{a, b, c\}$ de todas aquellas palabras que verifiquen simultáneamente las siguientes condiciones.

1. La palabra contiene un número par de a's.
2. La longitud de la palabra es un múltiplo de 3.
3. La palabra no contiene la subcadena *abc*.

La condición 1 se puede cumplir con el autómata de la Figura 1.34.
La condición 2 se puede cumplir con el autómata de la Figura 1.35.
La condición 3 se puede cumplir con el autómata de la Figura 1.36.
El autómata producto tiene los siguientes estados:

$$\begin{aligned} Q &= \{a_i b_j c_k \mid i \in \{0,1\}, j \in \{0,1,2\}, k \in \{0,1,2,3\}\} \\ F &= \{a_0b_0c_0, a_0b_0c_1, a_0b_0c_2\} \end{aligned}$$

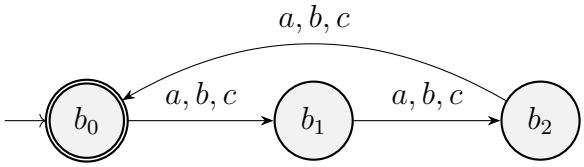


Figura 1.35: AFD que acepta palabras de longitud múltiplo de 3.

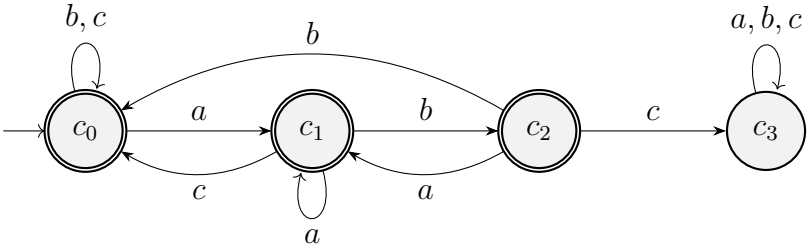


Figura 1.36: AFD que acepta palabras que no contienen la subcadena *abc*.

	1	2	3	4	5	6	7	8	9	10	11	12
δ	$a_0b_0c_0$	$a_0b_0c_1$	$a_0b_0c_2$	$a_0b_0c_3$	$a_0b_1c_0$	$a_0b_1c_1$	$a_0b_1c_2$	$a_0b_1c_3$	$a_0b_2c_0$	$a_0b_2c_1$	$a_0b_2c_2$	$a_0b_2c_3$
a	$a_1b_1c_1$	$a_1b_1c_1$	$a_1b_1c_1$	$a_1b_1c_3$	$a_1b_2c_1$	$a_1b_2c_1$	$a_1b_2c_1$	$a_1b_2c_3$	$a_1b_0c_1$	$a_1b_0c_1$	$a_1b_0c_1$	$a_1b_0c_3$
b	$a_0b_1c_0$	$a_0b_1c_2$	$a_0b_1c_0$	$a_0b_1c_3$	$a_0b_2c_0$	$a_0b_2c_2$	$a_0b_2c_0$	$a_0b_2c_3$	$a_0b_0c_0$	$a_0b_0c_2$	$a_0b_0c_0$	$a_0b_0c_3$
c	$a_0b_1c_0$	$a_0b_1c_0$	$a_0b_1c_3$	$a_0b_1c_3$	$a_0b_2c_0$	$a_0b_2c_0$	$a_0b_2c_3$	$a_0b_2c_3$	$a_0b_0c_0$	$a_0b_0c_0$	$a_0b_0c_3$	$a_0b_0c_3$

	13	14	15	16	17	18	19	20	21	22	23	24
δ	$a_1b_0c_0$	$a_1b_0c_1$	$a_1b_0c_2$	$a_1b_0c_3$	$a_1b_1c_0$	$a_1b_1c_1$	$a_1b_1c_2$	$a_1b_1c_3$	$a_1b_2c_0$	$a_1b_2c_1$	$a_1b_2c_2$	$a_1b_2c_3$
a	$a_0b_1c_1$	$a_0b_1c_1$	$a_0b_1c_1$	$a_0b_1c_3$	$a_0b_2c_1$	$a_0b_2c_1$	$a_0b_2c_1$	$a_0b_2c_3$	$a_0b_0c_1$	$a_0b_0c_1$	$a_0b_0c_1$	$a_0b_0c_3$
b	$a_1b_1c_0$	$a_1b_1c_2$	$a_1b_1c_0$	$a_1b_1c_3$	$a_1b_2c_0$	$a_1b_2c_2$	$a_1b_2c_0$	$a_1b_2c_3$	$a_1b_0c_0$	$a_1b_0c_2$	$a_1b_0c_0$	$a_1b_0c_3$
c	$a_1b_1c_0$	$a_1b_1c_0$	$a_1b_1c_3$	$a_1b_1c_3$	$a_1b_2c_0$	$a_1b_2c_0$	$a_1b_2c_3$	$a_1b_2c_3$	$a_1b_0c_0$	$a_1b_0c_0$	$a_1b_0c_3$	$a_1b_0c_3$

Tabla 1.1: Transiciones del autómata producto para el ejercicio 1.3.9.

	1	2	3	5	6	7	9	10	11	13	14	15	17	18	19	21	22	23
2	×																	
3	~~(9, 10)~~	×																
5	×	×	×															
6	×	×	×	×														
7	×	×	×	×	×													
9	×	×	×	×	×	×												
10	×	×	×	×	×	×	×											
11	×	×	×	×	×	×	×	×										
13	×	×	×	×	×	×	×	×	×									
14	×	×	×	×	×	×	×	×	×	×								
15	×	×	×	×	×	×	×	×	×	×	×							
17	×	×	×	×	×	×	×	×	×	×	×	×						
18	×	×	×	×	×	×	×	×	×	×	×	×	×					
19	×	×	×	×	×	×	×	×	~~(21, 22)~~	×	×	×	×	×				
21	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×			
22	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×		
23	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	
c_3	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

Tabla 1.2: Tabla de minimización del autómata producto para el ejercicio 1.3.9.

El autómata producto es $M = (Q, \{a, b, c\}, \delta, a_0b_0c_0, F)$, donde δ viene dado por la Tabla 1.1. Además, todos sus estados son accesibles.

En primer lugar, vemos que los estados de la forma $a_ib_jc_3$ son indistinguibles, puesto que desde ellos no se puede llegar a un estado final. Por tanto, los agrupamos en un único estado c_3.

La minimización del autómata producto se muestra en la Tabla 1.2.

Por tanto, el autómata minimal que acepta el lenguaje del ejercicio 1.3.9 es M, pero unificando los estados previamente descritos en c_3. Debido al gran número de estados (19), el grafo de dicho autómata no se muestra.

Ejercicio 1.3.10. 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 1.37.

Convertimos el AFND en un AFD, mostrado en la Figura 1.38.

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 $w \in \{a, b\}^*$:

$$\delta(q_0q_1q_3, w) \in F \qquad \delta(q_0q_2q_3, w) \in F$$

Por tanto, notemos $q_F = \{q_0q_1q_3, q_0q_2q_3\}$. El AFD minimal es el de la Figura 1.39.

- q_F es distinguible del resto de estados por ser el único estado final.
- q_0q_1 y q_0q_2 son distinguibles, ya que leyendo un a:

$$\delta(q_0q_1, a) = q_F \in F \qquad \delta(q_0q_2, a) = q_0q_1 \notin F$$

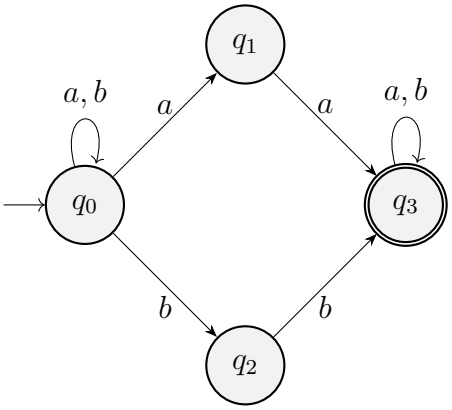


Figura 1.37: AFND asociado a la expresión regular $(a+b)^*(aa+bb)(a+b)^*$.

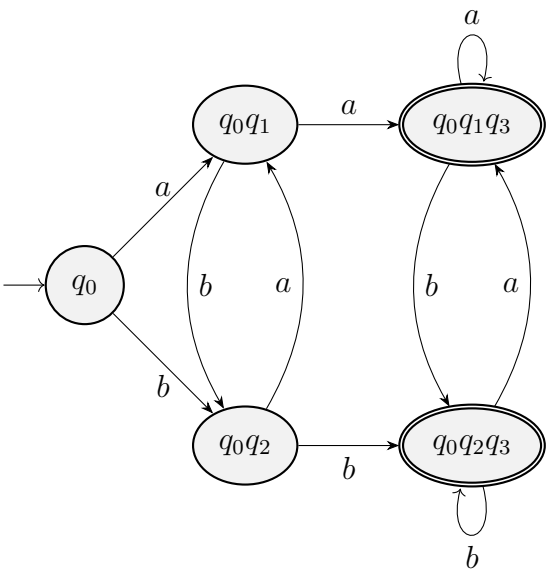


Figura 1.38: AFD asociado a la expresión regular $(a+b)^*(aa+bb)(a+b)^*$.

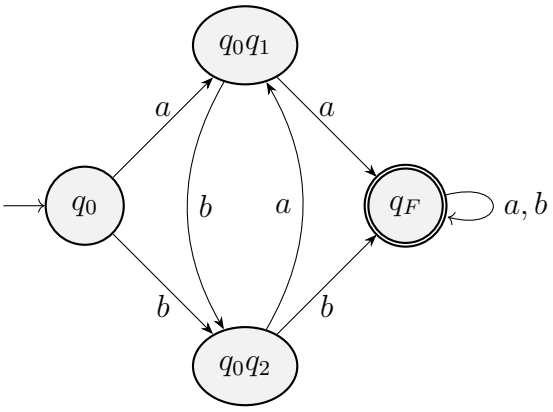


Figura 1.39: AFD minimal asociado a la expresión regular $(a+b)^*(aa+bb)(a+b)^*$.

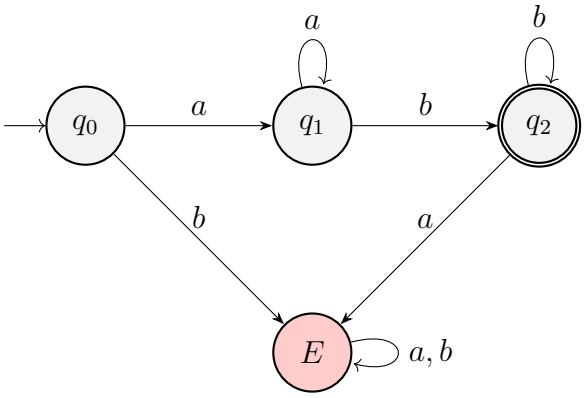


Figura 1.40: AFD asociado al lenguaje a^+b^+ del ejercicio 1.3.11.1.

- q_0 y q_0q_1 son distinguibles, ya que leyendo un a:

$$\delta(q_0, a) = q_0q_1 \notin F \qquad \delta(q_0q_1, a) = q_F \in F$$

- q_0 y q_0q_2 son distinguibles, ya que leyendo un b:

$$\delta(q_0, b) = q_0q_2 \notin F \qquad \delta(q_0q_2, b) = q_F \in F$$

Por tanto, el AFD minimal es el de la Figura 1.39.

Ejercicio 1.3.11. 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 1.40.

   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 distinguibles, ya que leyendo un b:

$$\delta(q_0, b) = E \notin F \qquad \delta(q_1, b) = q_2 \in F$$

   - q_0 y E son distinguibles, ya que leyendo un ab:

$$\delta^*(q_0, ab) = q_2 \in F \qquad \delta^*(E, ab) = E \notin F$$

   - q_1 y E son distinguibles, ya que leyendo un b:

$$\delta(q_1, b) = q_2 \in F \qquad \delta(E, b) = E \notin F$$

   Por tanto, el AFD minimal es el de la Figura 1.40. La gramática regular que genera el lenguaje es $G = (\{q_0, q_1, q_2\}, \{a, b\}, P, \{q_0\})$ con P:

$$\begin{aligned} q_0 &\longrightarrow aq_1 \\ q_1 &\longrightarrow aq_1 \mid bq_2 \\ q_2 &\longrightarrow bq_2 \mid \varepsilon \end{aligned}$$

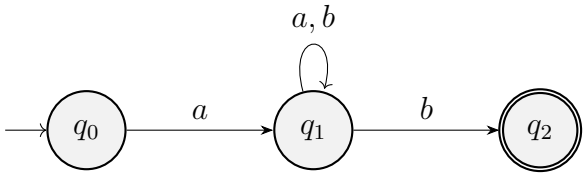


Figura 1.41: AFND asociado al lenguaje $a(a+b)^*b$ del ejercicio 1.3.11.2.

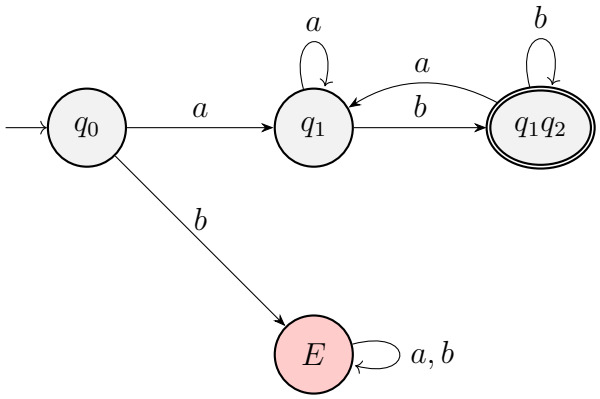


Figura 1.42: AFD asociado al lenguaje $a(a+b)^*b$ del ejercicio 1.3.11.2.

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

   En primer lugar, construimos el AFND asociado al lenguaje, mostrado en la Figura 1.41.

   Convertimos el AFND en un AFD, mostrado en la Figura 1.42.

   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 distinguibles, ya que leyendo un b:

   $$\delta(q_0, b) = E \notin F \qquad \delta(q_1, b) = q_1q_2 \in F$$

   - q_0 y E son distinguibles, ya que leyendo un ab:

   $$\delta^*(q_0, ab) = q_1q_2 \in F \qquad \delta^*(E, ab) = E \notin F$$

   - q_1 y E son distinguibles, ya que leyendo un b:

   $$\delta(q_1, b) = q_1q_2 \in F \qquad \delta(E, b) = E \notin F$$

   Por tanto, el AFD minimal es el de la Figura 1.42. La gramática regular que genera el lenguaje es $G = (\{q_0, q_1, q_2\}, \{a, b\}, P, \{q_0\})$ con P:

   $$\begin{aligned} q_0 &\longrightarrow aq_1 \\ q_1 &\longrightarrow aq_1 \mid bq_2 \\ q_2 &\longrightarrow bq_2 \mid aq_1 \mid \varepsilon \end{aligned}$$

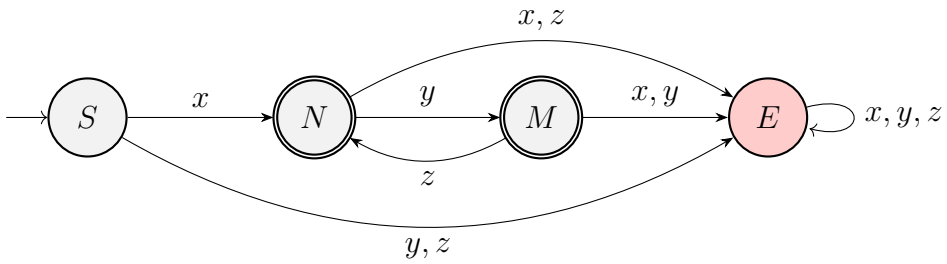


Figura 1.43: AFD asociado al lenguaje $\mathcal{L}(G)$ del ejercicio 1.3.12.

Ejercicio 1.3.12. Considera la gramática cuyas producciones se presentan a continuación y donde el símbolo inicial es S:

$$\begin{aligned} S &\to xN \mid x \\ N &\to yM \mid y \\ M &\to zN \mid z \end{aligned}$$

1. Escribe el diagrama de transiciones para el AFD que acepte el lenguaje $\mathcal{L}(G)$ generado por G.

   Las siguientes producciones generan el mismo lenguaje:

   $$\begin{aligned} S &\to xN \\ N &\to yM \mid \varepsilon \\ M &\to zN \mid \varepsilon \end{aligned}$$

   Por tanto, el AFD asociado al lenguaje $\mathcal{L}(G)$ es el de la Figura 1.43.

2. Encuentra una gramática regular por la izquierda que genere ese mismo lenguaje $\mathcal{L}(G)$.

   La expresión regular asociada al lenguaje $\mathcal{L}(G)$ es:

   $$x(yz)^*(y + \varepsilon)$$

   Sea $G' = (\{S, N, M, F\}, \{x, y, z\}, P, F)$ con P:

   $$\begin{aligned} F &\to N \mid M \\ N &\to Sx \mid Mz \\ M &\to Ny \\ S &\to \varepsilon \end{aligned}$$

   Tenemos que G' es una gramática regular por la izquierda, y $\mathcal{L}(G') = \mathcal{L}(G)$.

3. Encuentra el AFD que acepte el complementario del lenguaje $\mathcal{L}(G)$.

   Intercambiando los estados finales por no finales y viceversa en el AFD de la Figura 1.43, obtenemos el AFD asociado al complementario del lenguaje $\mathcal{L}(G)$, mostrado en la Figura 1.44.

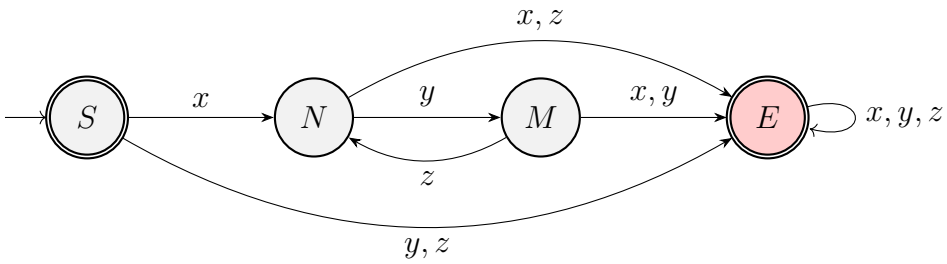


Figura 1.44: AFD asociado al lenguaje $\overline{\mathcal{L}(G)}$ del ejercicio 1.3.12.

q_1	×		
q_2	×	×	
q_3	×	(q_0, q_3)	×
	q_0	q_1	q_2

Tabla 1.3: Tabla de minimización de M_1.

Ejercicio 1.3.13. Determinar autómatas minimales para los lenguajes $L(M_1) \cup L(M_2)$ y $L(M_1) \cap \overline{L(M_2)}$ donde,

- $M_1 = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \delta_1, q_0, \{q_2\})$ 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

- $M_2 = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \delta_2, q_0, \{q_2\})$ 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 miminización de M_1 se muestra en la Tabla 1.3.

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 miminización de M_2 se muestra en la Tabla 1.4.

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 $L(M_1) \cap \overline{L(M_2)}$.

q_1	×		
q_2	×	×	
q_3	×	×	×
	q_0	q_1	q_2

Tabla 1.4: Tabla de minimización de M_2.

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_1q_1'	q_1q_1'	q_1q_3'	q_1q_3'	q_1q_1'	q_1q_1'	q_1q_3'	q_1q_3'	q_1q_1'	q_1q_1'	q_1q_3'	q_1q_3'
b	q_2q_1'	q_2q_2'	q_2q_2'	q_2q_3'	q_1q_1'	q_1q_2'	q_1q_2'	q_1q_3'	q_1q_1'	q_1q_2'	q_1q_2'	q_1q_3'
c	q_1q_3'	q_1q_3'	q_1q_0'	q_1q_3'	q_1q_3'	q_1q_3'	q_1q_0'	q_1q_3'	q_0q_3'	q_0q_3'	q_0q_0'	q_0q_3'

Tabla 1.5: Transiciones del autómata producto para $L(M_1),\ L(M_2)$.

1. $L(M_1) \cup L(M_2)$

   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:

   - $Q = \{q_iq_j' \mid i = 0,1,2 \text{ y } j = 0,1,2,3\}$.
   - $F = \{(q_2,q_j') \mid j = 0,1,2,3\} \cup \{(q_i,q_2') \mid i = 0,1,2\}$.
   - δ viene dada por la Tabla 1.5.

   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 1.6.

   Por tanto, el AFD minimal asociado a $L(M_1) \cup L(M_2)$ es $M^m = M$, el cual se puede ver en la Figura 1.45.

2. $L(M_1) \cap \overline{L(M_2)}$

   Ya tenemos del apartado anterior el AFD minimal asociado a $L(M_1)$. La minimización de $\overline{L(M_2)}$ se muestra en la Tabla 1.7.

   Por tanto, el AFD minimal asociado a $\overline{L(M_2)}$ es ya minimal. A continuación, construimos el AFD asociado a $L(M_1) \cap \overline{L(M_2)}$. Sea este $M = (Q, \{a,b,c\}, \delta, (q_0,q_0'), F)$, donde:

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'	q_2q_1'

Tabla 1.6: Tabla de minimización de M para $L(M_1) \cup L(M_2)$.

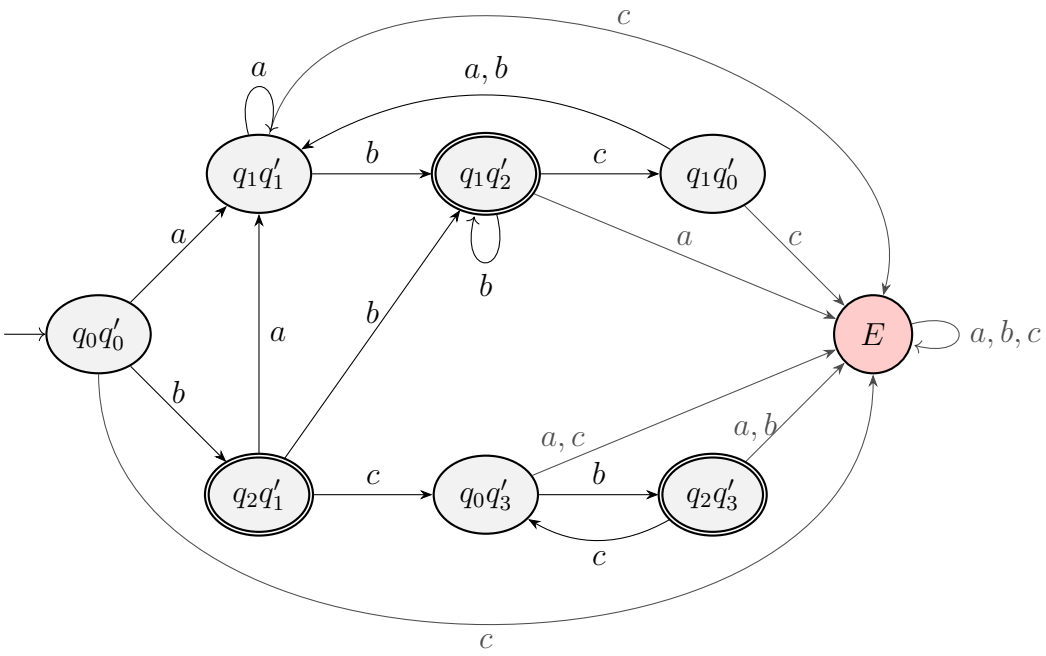


Figura 1.45: AFD minimal asociado a $L(M_1) \cup L(M_2)$.

q_1	×		
q_2	×	×	
q_3	×	×	×
	q_0	q_1	q_2

Tabla 1.7: Tabla de minimización de $\overline{M_2}$.

	q_0q_0'	q_0q_3'	q_1q_0'	q_1q_1'	q_1q_2'	q_1q_3'	q_2q_1'
q_0q_3'							
q_1q_0'	×	×					
q_1q_1'	×	×					
q_1q_2'	×	×		(q_1q_2', q_1q_0') (q_qq_1', q_1q_0')			
q_1q_3'	×	×	(q_1q_2', q_1q_3') (q_1q_2', q_1q_0') (q_1q_1', q_1q_2') $(q0q_3', q_1q_0')$	(q_2q_3', q_2q_1') (q_1q_3', q_1q_0') (q_1q_1', q_1q_2')	(q_2q_3', q_2q_1') (q_1q_1', q_1q_3')		
q_2q_1'	×	×	×	×	×	×	
q_2q_3'	×	×	×	×	×	×	(q_0q_3', q_0q_0')

Tabla 1.8: Tabla de minimización de M para $L(M_1) \cap \overline{L(M_2)}$.

δ^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

Tabla 1.9: Transiciones del autómata minimal para $L(M_1),\ \overline{L(M_2)}$.

- $Q = \{q_iq_j' \mid i = 0, 1, 2 \text{ y } j = 0, 1, 2, 3\}$.
- $F = \{(q_2, q_j') \mid j = 0, 1, 3\}$.
- δ viene dada por la Tabla 1.5.

Los estados accesibles son los mismos que antes:

$$\{q_0q_0', q_0q_3', q_1q_0', q_1q_1', q_1q_2', q_1q_3', q_2q_1', q_2q_3'\}$$

La minimización de M se muestra en la Tabla 1.8.

Por tanto, notando por $\equiv$ a la relación de indistinguibilidad, tenemos que:

$$q_0q_3' \equiv q_0q_0' := q_0 \qquad q_1q_1' \equiv q_1q_0' \equiv q_1q_2' \equiv q_1q_3' := q_1 \qquad q_2q_1' \equiv q_2q_3' := q_2$$

Por tanto, el AFD minimal asociado a $L(M_1) \cap \overline{L(M_2)}$ es $M^m = (Q, \{a, b, c\}, \delta^m, q_0, F)$ donde:

- $Q = \{q_0, q_0q_1', q_0q_2', q_1, q_2, q_2q_0', q_2q_2'\}$.
- $F = \{q_2, q_2q_0'\}$.
- Los estados accesibles son $\{q_0, q_1, q_2\}$.
- δ^m viene dada por la Tabla 1.9, donde solo la definimos para los estados accesibles.

El AFD minimal asociado a $L(M_1) \cap \overline{L(M_2)}$ es el de la Figura 1.46.

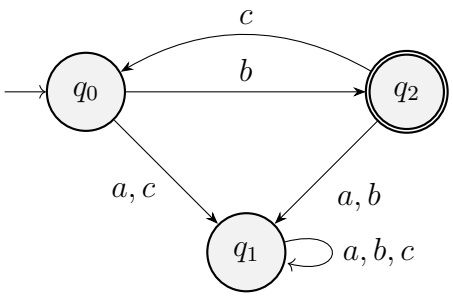


Figura 1.46: AFD minimal asociado a $L(M_1) \cap \overline{L(M_2)}$.

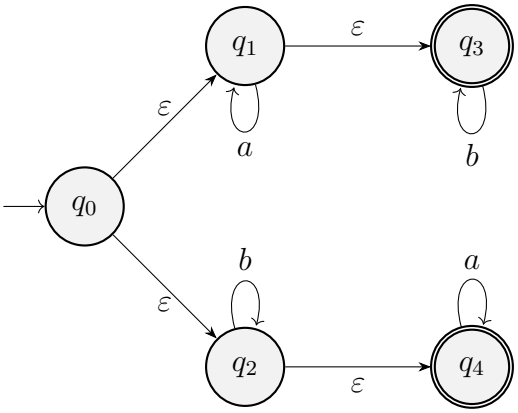


Figura 1.47: AFND con transiciones nulas asociado a la expresión regular $a^*b^* + b^*a^*$.

Ejercicio 1.3.14. Dado el conjunto regular representado por la expresión regular $a^*b^* + b^*a^*$, construir un autómata finido determinístico minimal que lo acepte.

El AFND con transiciones nulas asociado a la expresión regular dada es el de la Figura 1.47.

El AFD asociado a la expresión regular dada es el de la Figura 1.48.

Veamos ahora que el AFD de la Figura 1.48 es minimal.

- El estado E es distinguible de cualquier otro estado pues ser el único estado que no es final.

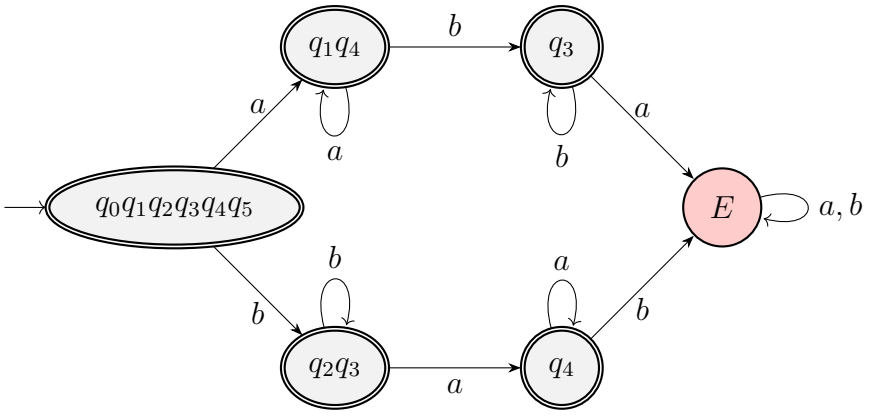


Figura 1.48: AFD asociado a la expresión regular $a^*b^* + b^*a^*$.

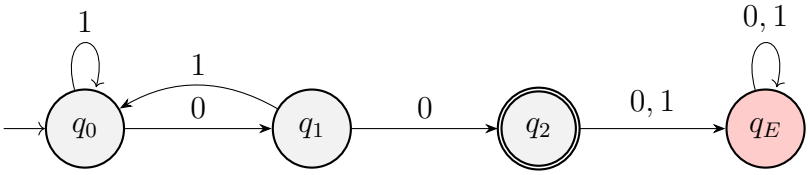


Figura 1.49: AFD minimal asociado a $L_1 = (01+1)^*00$.

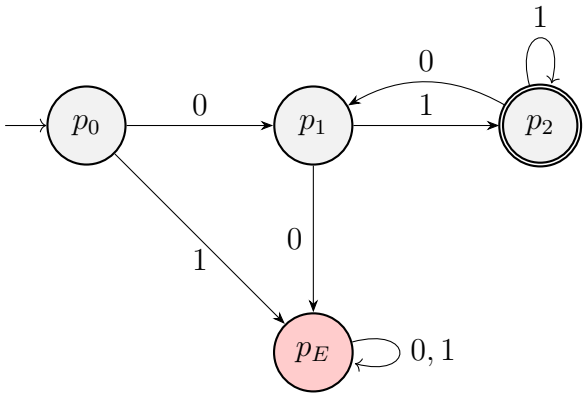


Figura 1.50: AFD minimal asociado a $L_2 = 01(01+1)^*$.

- Veamos que q_3 es distinguible del resto de estados finales. Leyendo a, tenemos que:

$$\delta(q_3, a) = E \notin F, \qquad \begin{cases} \delta(q_1q_4, a) = q_1q_4 \in F, \\ \delta(q_2q_3, a) = q_4 \in F, \\ \delta(q_4, a) = q_4 \in F, \\ \delta(q_0q_1q_2q_3q_4q_5, a) = q_1q_4 \in F. \end{cases}$$

- De forma análoga leyendo b, tenemos que q_4 es distinguible del resto de estados finales.
- El estado q_1q_4 es distinguible de q_2q_3 y $q_0q_1q_2q_3q_4q_5$ leyendo ba.
- Finalmente, $q_0q_1q_2q_3q_4q_5$ es distinguible de q_2q_3 leyendo ab.

Ejercicio 1.3.15. 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.

Veamos que $L_1 \cap L_2 = \emptyset$. Sea $z \in L_1$. Entonces, z es admitida por la expresión regular $(01+1)^*00$, por lo que termina en 0. Por tanto, $z \notin L_2$, ya que todas las palabras de L_2 terminan por 1. Por tanto, $L_1 \cap L_2 = \emptyset$ y $L_1 \setminus L_2 = L_1$. Aun así, haremos el proceso algorítmico para obtener el AFD minimal asociado a $L_1 \setminus L_2$.

El AFD asociado a L_1 es el de la Figura 1.49.
El AFD asociado a L_2 es el de la Figura 1.50.

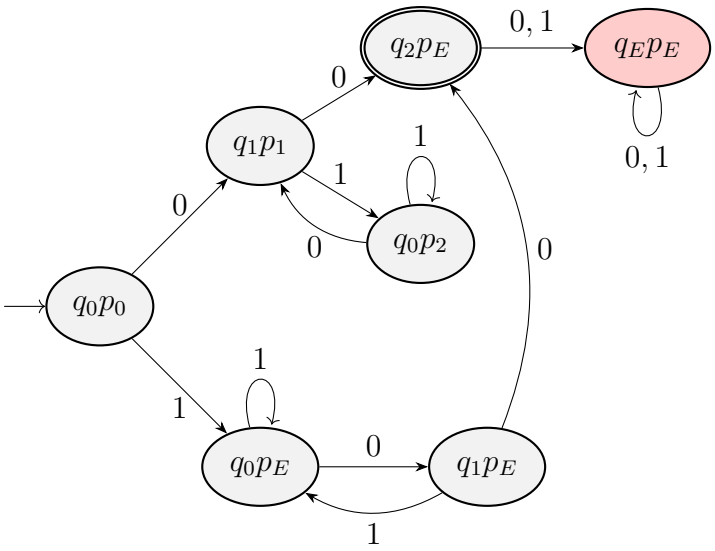


Figura 1.51: AFD asociado a $L_1 \setminus L_2$.

q_1p_1	×				
q_2p_E	×	×			
q_0p_2		×	×		
q_0p_E		×	×	$(q_1p_E,\ q_1p_1)$ (q_0p_0, q_0p_2)	
q_1p_E	×	(q_0p_E, q_0p_0) (q_0p_E, q_0p_2)	×	×	×
q_Ep_E	×	×	×	×	×	×
	q_0p_0	q_1p_1	q_2p_E	q_0p_2	q_0p_E	q_1p_E

Tabla 1.10: Tabla de minimización de $L_1 \setminus L_2$.

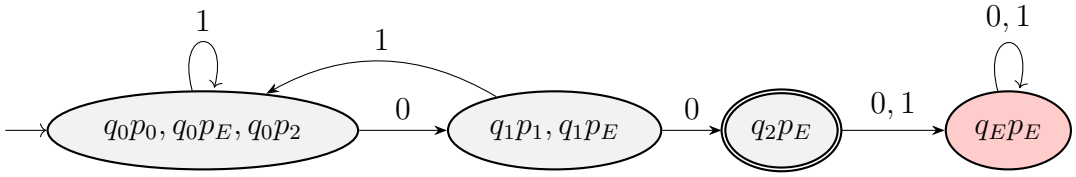


Figura 1.52: AFD minimal asociado a $L_1 \setminus L_2$.

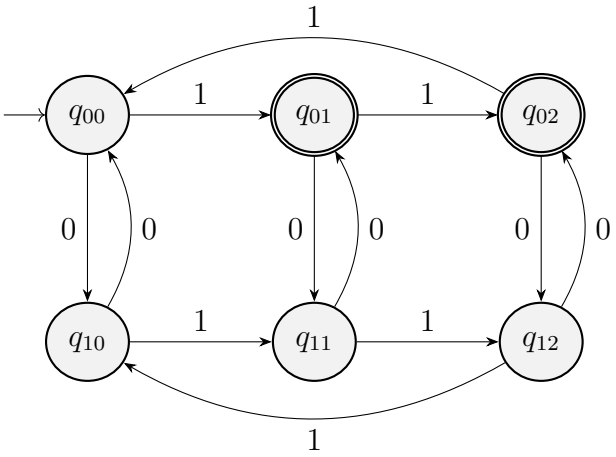


Figura 1.53: AFD que acepta el lenguaje L del Ejercicio 1.3.16.

El AFD asociado a $L_1 \setminus L_2$ es el de la Figura 1.51.
La minimización del AFD de la Figura 1.51 se muestra en la Tabla 1.10.
Por tanto, notando por $\equiv$ a la relación de indistinguibilidad, tenemos que:

$$q_0p_0 \equiv q_0p_E \equiv q_0p_2 \qquad q_1p_1 \equiv q_1p_E$$

El AFD minimal asociado a $L_1 \setminus L_2$ es el de la Figura 1.52. Como vemos, este autómata es isomorfo al autómata de la Figura 1.49 que aceptaba L_1, como ya habíamos predicho.

Ejercicio 1.3.16. Dados los alfabetos $A = \{0, 1, 2, 3\}$ y $B = \{0, 1\}$ y el homomorfismo f de A^* en B^* dado por:

$$f(0) = 00, \quad f(1) = 01, \quad f(2) = 10, \quad f(3) = 11$$

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 $f^{-1}(L)$.

El AFD que acepta a L se desarrolló en el Ejercicio 1.2.17. Se muestra no obstante de nuevo en la Figura 1.53.
El AFD que acepta a $f^{-1}(L)$ es el de la Figura 1.54.

Ejercicio 1.3.17. Determinar un autómata finito determinístico minimal para el lenguaje sobre el alfabeto $A = \{a, b, c\}$ dado por la expresión regular $b(a + b)^* + cb^*$.

El AFD asociado a la expresión regular dada es el de la Figura 1.55.
Veamos que es minimal:

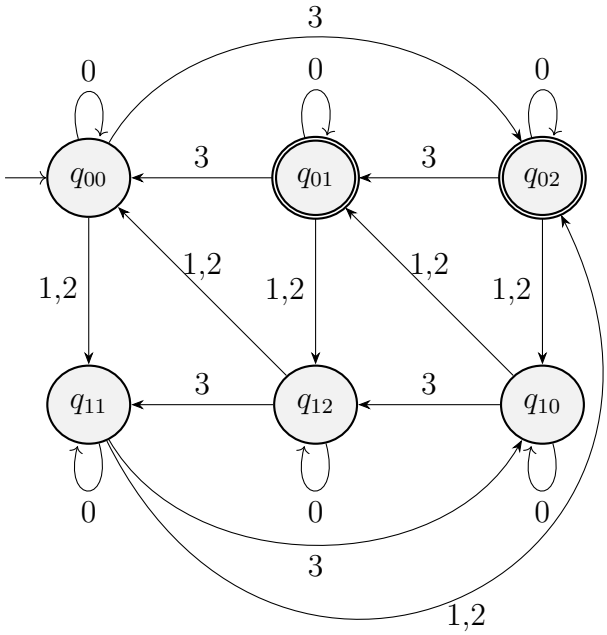


Figura 1.54: AFD que acepta el lenguaje L del Ejercicio 1.3.16.

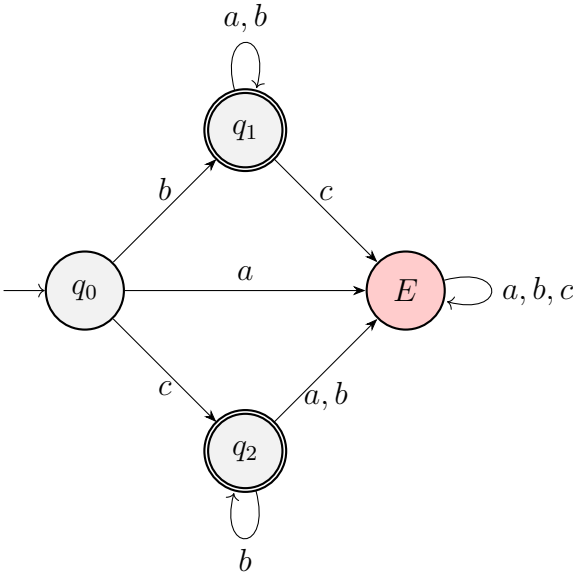


Figura 1.55: AFD asociado a la expresión regular $b(a+b)^* + cb^*$.

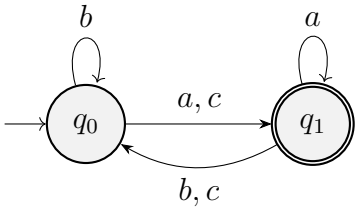


Figura 1.56: AFD minimal asociado a $L_1 = L_3$ del Ejercicio 1.3.18.

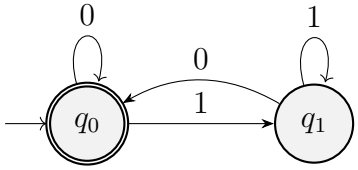


Figura 1.57: AFD minimal asociado a los múltiplos de 2 del Ejercicio 1.3.19.

- E es distinguible de cualquier otro estado puesto que desde él no se puede llegar a un estado final.
- q_0 es distinguible de q_1 y q_2 puesto que no es final.
- q_1 es distinguible de q_2 leyendo a.

Ejercicio 1.3.18. Determinar si las expresiones regulares siguientes representan el mismo lenguaje:

1. $L_1 = (b + (c + a)a^*(b + c))^*(c + a)a^*$
2. $L_2 = b^*(c + a)((b + c)b^*(c + a))^*a^*$
3. $L_3 = b^*(c + a)(a^*(b + c)b^*(c + a))^*a^*$

Justificar la respuesta.

Veamos en primer lugar que $L_2 \neq L_1, L_3$. Sea la palabra $u = caaccaaa$. Tenemos que $u \in L_1, L_3$ pero $u \notin L_2$. Por tanto, $L_2 \neq L_1, L_3$.

Veamos ahora que $L_1 = L_3$ obteniendo el autómata finito minimal asociado a cada uno de ellos y viendo que es igual. Este se encuentra en la Figura 1.56.

Ejercicio 1.3.19. Construir un autómata finito determinista minimal que acepte el conjunto de palabras sobre el alfabeto $A = \{0, 1\}$ que representen números no divisibles por dos ni por tres (en binario).

El AFD asociado a los múltiplos de 2 se muestra en la Figura 1.57.

El AFD asociado a los múltiplos de 3 se muestra en la Figura 1.58.

El autómata descrito en el enunciado es el autómata producto de los complementarios a los dos anteriores, mostrado en la Figura 1.59.

La minimización del autómata de la Figura 1.59 se muestra en la Tabla 1.11.

Por tanto, notando por $\equiv$ a la relación de indistinguibilidad, tenemos que:

$$q_0p_0 \equiv q_1p_0 \equiv: p_0$$

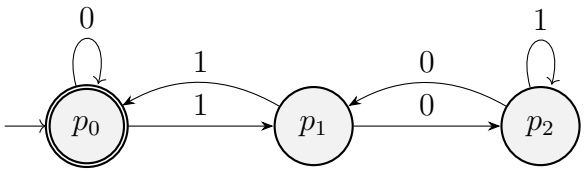


Figura 1.58: AFD minimal asociado a los múltiplos de 3 del Ejercicio 1.3.19.

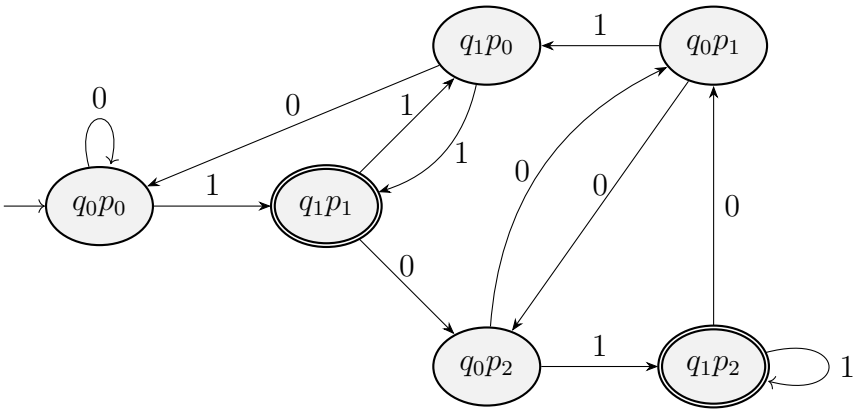


Figura 1.59: AFD asociado al lenguaje del Ejercicio 1.3.19.

q_1p_1	×				
q_1p_0		×			
q_0p_2	×	×	×		
q_0p_1	×	×	×	×	
q_1p_2	×	×	×	×	×
	q_0p_0	q_1p_1	q_1p_0	q_0p_2	q_1p_1

Tabla 1.11: Tabla de minimización del autómata de la Figura 1.59.

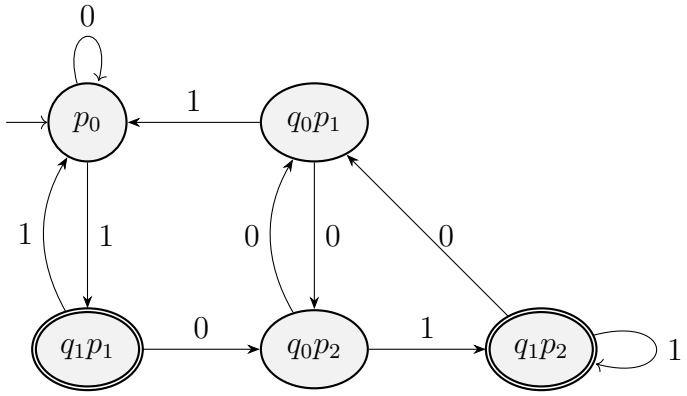


Figura 1.60: AFD Minimal asociado al lenguaje del Ejercicio 1.3.19.

Por tanto, el autómata minimal asociado al lenguaje del enunciado es el de la Figura 1.60.

Ejercicio 1.3.20. Determinar una expresión regular para los siguientes lenguajes sobre el alfabeto $\{0, 1\}$:

- Palabras en las que el tercer símbolo es un 0.
- Palabras en las que el antepenúltimo símbolo es un 1.

Construir un autómata finito minimal que acepte la intersección de ambos lenguajes.

Respecto al leguaje de las palabras en las que el tercer símbolo es un 0 (llamémoslo L_1), su expresión regular es:

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

Respecto al leguaje de las palabras en las que el antepenúltimo símbolo es un 1 (llamémoslo L_2), su expresión regular es:

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

Opción 1: Autómata Producto.

Obtendremos el AFD de L_1 de forma directa, mostrado en la Figura 1.61.

Podríamos optar con construir el AFD de L_2 de forma directa, pero construiremos un AFND que acepte el lenguaje y luego lo convertiremos a AFD. El AFND se muestra en la Figura 1.62, cuyos estados son:

- q_0: No estamos en la cadena final, por lo que podemos leer 0's y 1's.
- q_1: Acabo de empezar la cadena final. He leído un 1.
- q_2: Estoy en la cadena final. El leído el 1 y el segundo símbolo.
- q_3: Hemos terminado la cadena final.

Convertimos ahora el AFND de la Figura 1.62 en un AFD, representado en la Figura 1.63.

El AFD de la Figura 1.63 acepta el lenguaje L_2, y es idéntico al que podríamos haber razonado de forma directa. Veamos qué representa cada estado:

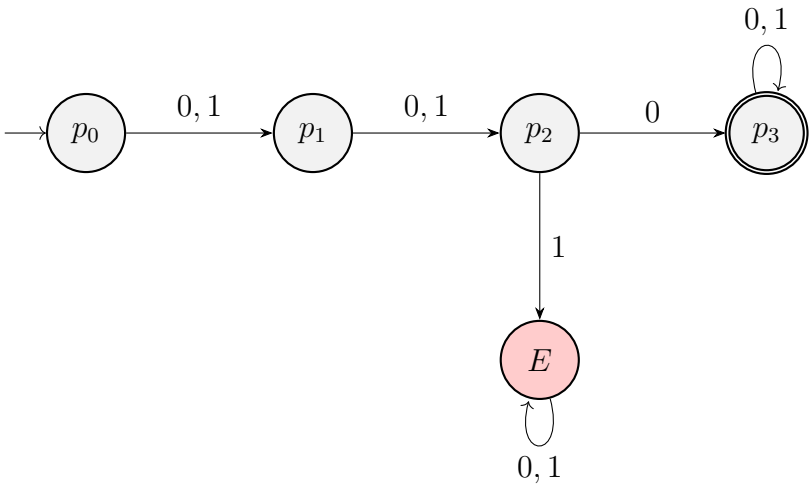


Figura 1.61: AFD minimal que acepta el lenguaje L_1 del ejercicio 1.3.20.

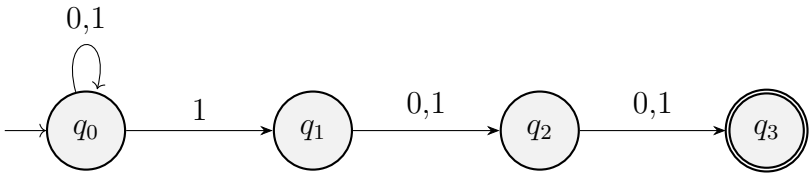


Figura 1.62: AFND que acepta el lenguaje L_2 del ejercicio 1.3.20.

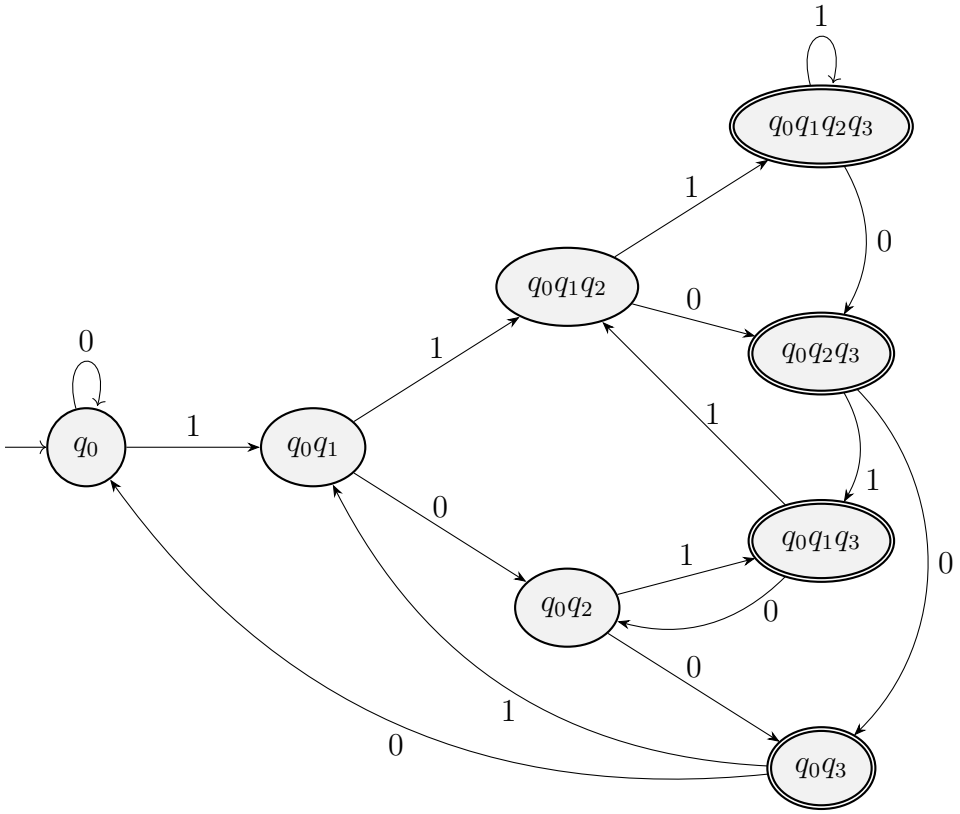


Figura 1.63: AFD que acepta el lenguaje L_2 del ejercicio 1.3.20.

	q_0	q_0q_1	$q_0q_1q_2$	q_0q_2	$q_0q_1q_2q_3$	$q_0q_2q_3$	$q_0q_1q_3$
q_0q_1	×						
$q_0q_1q_2$	×	~~$(q_0q_3, q_0q_1q_3)$~~					
q_0q_2	~~$(q_0q_3, q_0q_1q_3)$~~	×	×				
$q_0q_1q_2q_3$	×	×	×	×			
$q_0q_2q_3$	×	×	×	×	×		
$q_0q_1q_3$	×	×	×	×	×	×	
q_0q_3	×	×	×	×	×	×	×

Tabla 1.12: Tabla de minimización del autómata de la Figura 1.63.

- q_0: No estamos en un candidado a ser cadena final. Si leemos un 1, empezaremos la que puede ser la cadena final.
- q_0q_1: Hemos leído un 1, por lo que hemos empezado la posible cadena final. Llevamos 1.
- $q_0q_1q_2$: Hemos leído un 1 y un 1. Llevamos 11 de cadena final.
- q_0q_2: Hemos leído un 1 y un 0. Llevamos 10 de cadena final.
- $q_0q_1q_2q_3$: Hemos leído un 1, un 1 y un 1. Llevamos 111 de cadena final.
- $q_0q_2q_3$: Hemos leído un 1, un 1 y un 0. Llevamos 110 de cadena final.
- $q_0q_1q_3$: Hemos leído un 1, un 0 y un 1. Llevamos 101 de cadena final.
- q_0q_3: Hemos leído un 1, un 0 y un 0. Llevamos 100 de cadena final.

Lo complejo de hacerlo de forma directa sería ver las transiciones desde los estados finales. Razonando cuál es la cadena final leída, podríamos haberlo hecho de forma directa, pero el AFND nos ha ayudado a hacerlo de forma algorítmica. La minimización del autómata de la Figura 1.63 se muestra en la Tabla 1.12, donde vemos que este era minimal.

Como vemos, obtener el autómata producto será complejo, puesto que tendrá $8 \cdot 5 = 40$ estados. Por tanto, optamos por la segunda opción.

Opción 2: Expresión Regular.

La expresión regular del lenguaje intersección ha de ser la siguiente:

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

Observación. Notemos que esta expresión regular no contempla las palabras de longitud menor o igual a 5. Estudiemos estas:

- Si $|z| = 5$, es necesario que el tercer símbolo sea un 0 y el antepenúltimo $(5-2=3)$ un 1. Por tanto, como el tercer símbolo no puede tomar ambos valores a la vez, no hay palabras de longitud 5.
- Si $|z| = 4$, es necesario que el tercer símbolo sea un 0 y el antepenúltimo $(4-2=2)$ un 1. Por tanto, estas palabras son de la forma $(0+1)\ 10\ (0+1)$.
- Si $|z| = 3$, es necesario que el tercer símbolo sea un 0 y el antepenúltimo $(3-2=1)$ un 1. Por tanto, estas palabras son de la forma $1\ (0+1)\ 0$.

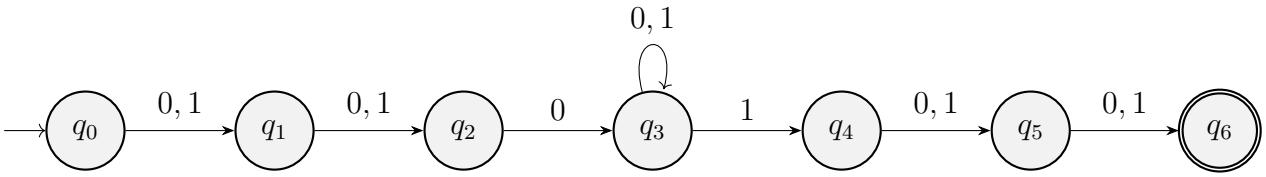


Figura 1.64: AFND que acepta la intersección de los lenguajes del ejercicio 1.3.20.

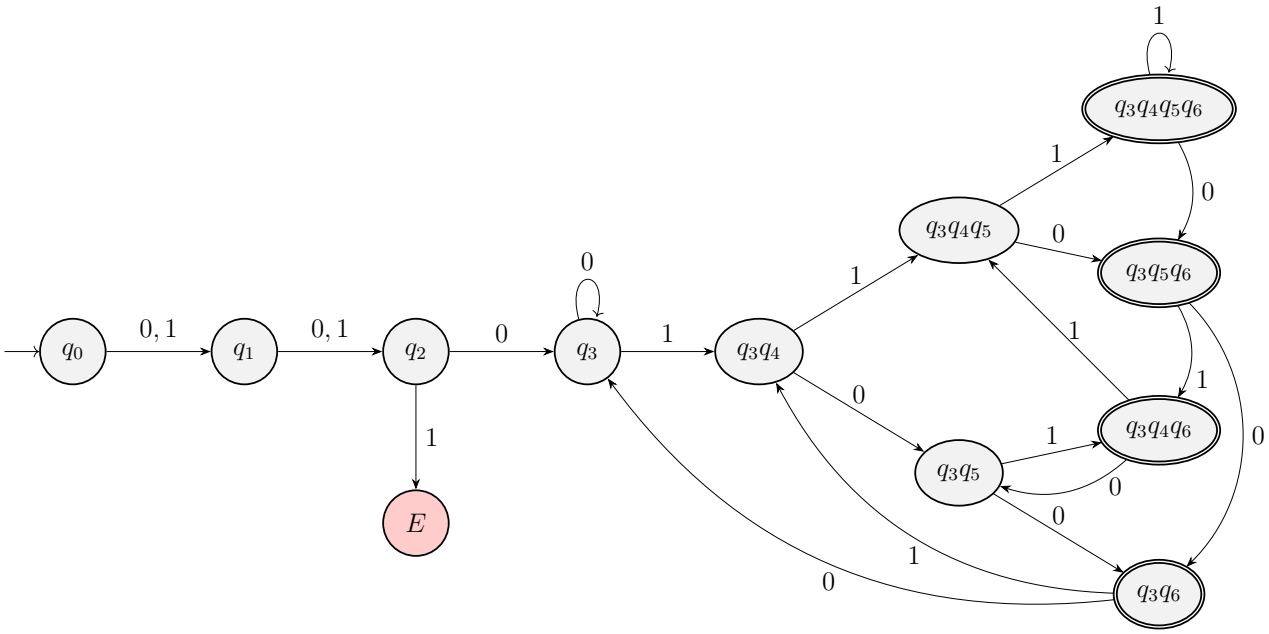


Figura 1.65: AFD que acepta la intersección de los lenguajes del ejercicio 1.3.20.

- Si $|z| < 3$, no podemos considerar el tercer símbolo, por lo que no hay palabras de longitud menor que 3.

Por tanto, la expresión regular correcta sería:

$$(0+1)(0+1)\ 0\ (0+1)^*\ 1\ (0+1)(0+1) + (0+1)\ 10\ (0+1) + 1\ (0+1)\ 0$$

No obstante, obtener el autómata finito minimal asociado a esta expresión regular sería muy complejo, por lo que no consideramos dichas palabras.

El AFND que acepta el lenguaje intersección se muestra en la Figura 1.64.

Convertimos ahora el AFND de la Figura 1.64 en un AFD, representado en la Figura 1.65.

Ejercicio 1.3.21. Construir un autómata finito minimal para los siguientes lenguajes sobre el alfabeto $\{0, 1\}$:

1. Palabras que contienen como subcadena una palabra del conjunto $\{00, 11\}^2$.

   Han de contener una subcadena del conjunto $\{0000, 0011, 1100, 1111\}$. Para ello, construimos el autómata de la Figura 1.66.

   La minimización del autómata de la Figura 1.66 se muestra en la Tabla 1.13.

   Por tanto, tenemos que todos los estados finales eran indistinguibles. Notándolos por q_F, el autómata minimal es el de la Figura 1.67.

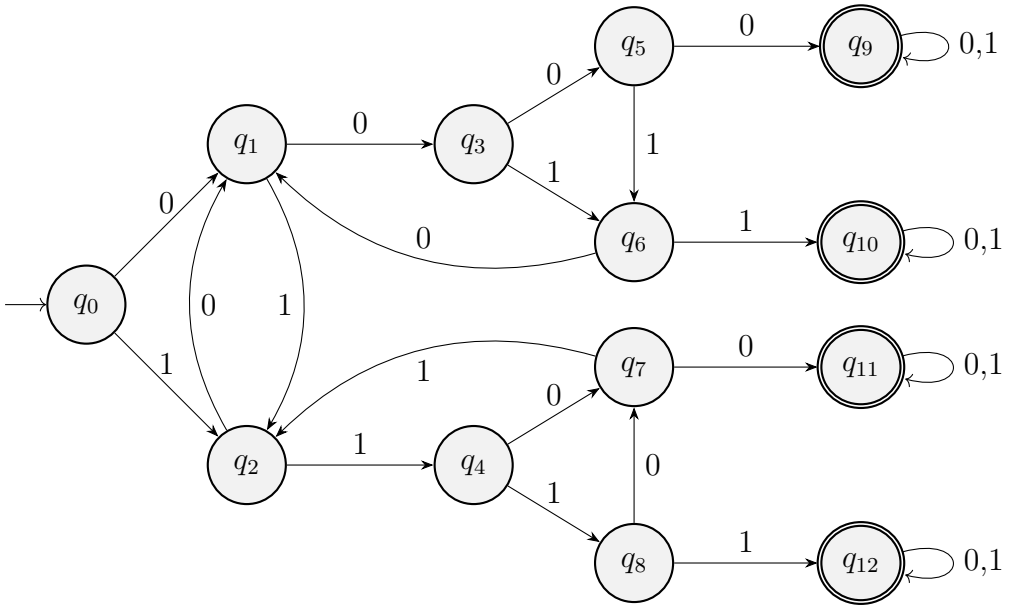


Figura 1.66: AFD que acepta el lenguaje del ejercicio 1.3.21.1.

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}	q_{11}
q_1	×											
q_2	×	×										
q_3	×	×	×									
q_4	×	×	×	×								
q_5	×	×	×	×	×							
q_6	×	×	×	×	×	×						
q_7	×	×	×	×	×	×	×					
q_8	×	×	×	×	×	×	×	×				
q_9	×	×	×	×	×	×	×	×	×			
q_{10}	×	×	×	×	×	×	×	×	×			
q_{11}	×	×	×	×	×	×	×	×	×			
q_{12}	×	×	×	×	×	×	×	×	×			

Tabla 1.13: Tabla de minimización del autómata de la Figura 1.66.

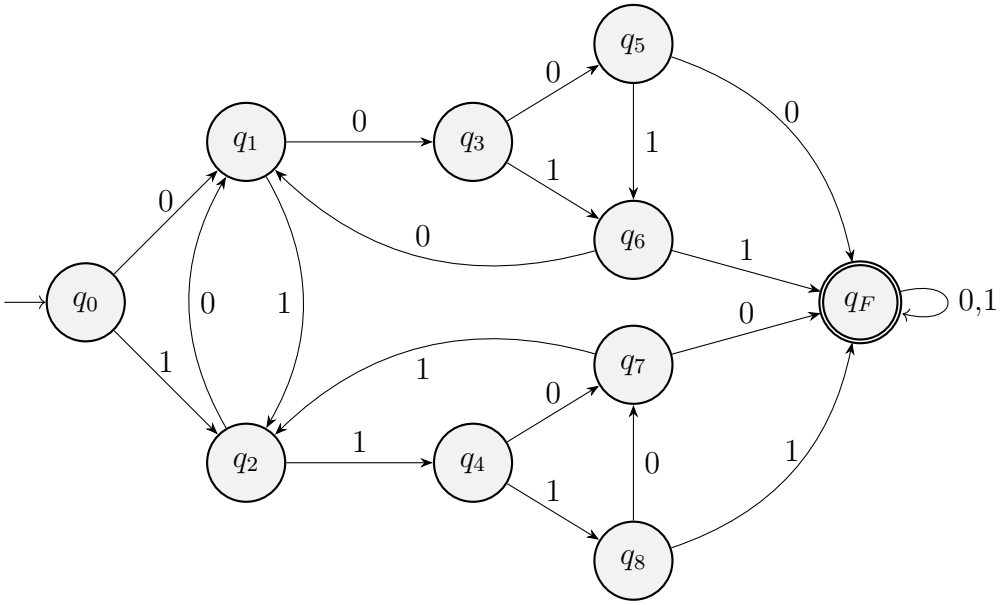


Figura 1.67: AFD minimal que acepta el lenguaje del ejercicio 1.3.21.1.

2. Palabras que contienen como subcadena una palabra del conjunto $\{0011, 1100\}$.

   El AFD minimal que acepta este lenguaje es el de la Figura 1.68.

Ejercicio 1.3.22. Responda a los siguientes apartados:

1. Construye una gramática regular que genere el siguiente lenguaje:

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

   Este es muy similar al Ejercicio 1.2.15. Tenemos los siguientes estados:

   - $\underline{S}$: La cadena leída es correcta, 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 impar. El número de 1's es par.
   - $\underline{E_1}$: Tenemos un error en 1, ya que el número de 1's es impar. El número de 0's es par.
   - $\underline{E_{01}}$: Tenemos un error en 0 y en 1, ya que el número de 0's y de 1's es impar.

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

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

   El autómata finito determinista minimal asociado a la gramática obtenida es el de la Figura 1.69.

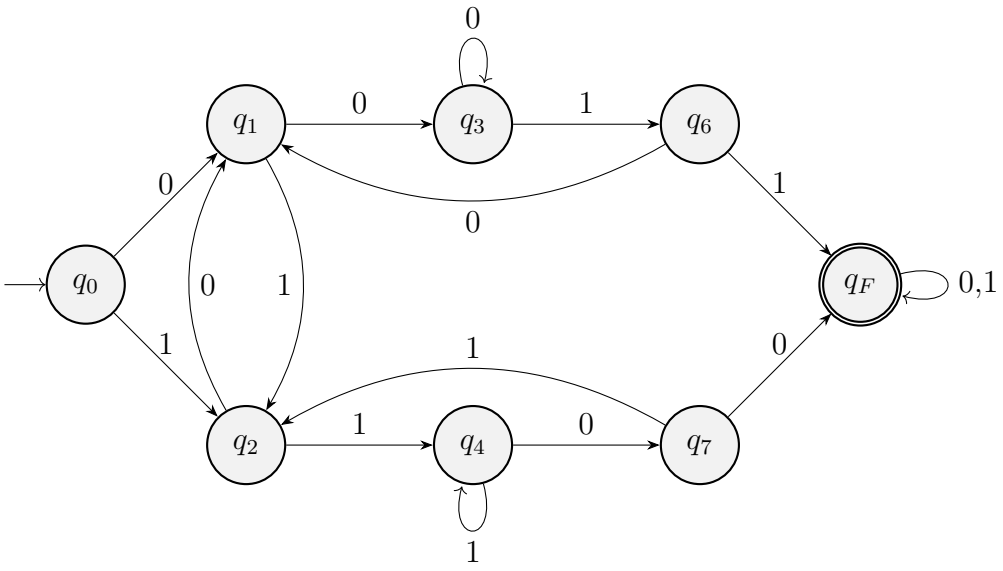


Figura 1.68: AFD minimal que acepta el lenguaje del ejercicio 1.3.21.2.

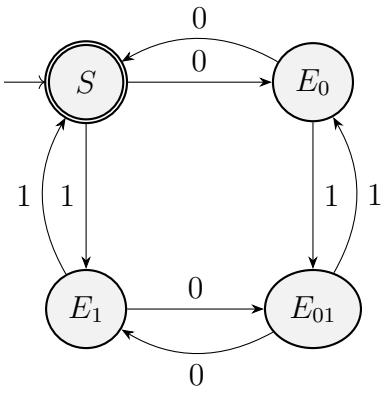


Figura 1.69: AFD minimal asociado a L_1 del Ejercicio 1.3.22.

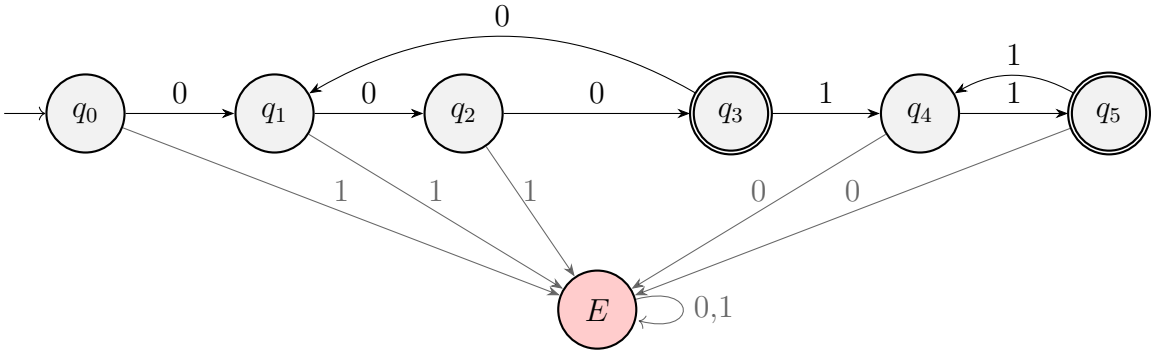


Figura 1.70: AFD minimal asociado a L_2 del Ejercicio 1.3.22.

2. Construye un autómata que reconozca el siguiente lenguaje:

$$L_2 = \{0^n 1^m \mid n \geqslant 1, m \geqslant 0, n \text{ múltiplo de 3, } m \text{ par } \}$$

Sean los siguientes estados:

- $\underline{q_0}$: $n, m = 0$.
- $\underline{q_1}$: $n \mod 3 = 1$, $m = 0$.
- $\underline{q_2}$: $n \mod 3 = 2$, $m = 0$.
- $\underline{q_3}$: $n \mod 3 = 0$, $n > 1$, $m = 0$.
- $\underline{q_4}$: $n \mod 3 = 0$, $m \mod 2 = 1$.
- $\underline{q_5}$: $n \mod 3 = 0$, $m \mod 2 = 0$.

El autómata finito determinista asociado a L_2 es el de la Figura 1.70.

3. Diseña el AFD mínimo que reconoce el lenguaje $(L_1 \cup L_2)$.

Ejercicio 1.3.23. Sobre el alfabeto $\{0, 1\}$:

1. Construye una gramática regular que genere el lenguaje L_1 de las palabras u tales que:

- Si $|u| < 5$ entonces el número de 1's es impar.
- Si $|u| \geqslant 5$ entonces el número de 1's es par.
- u tiene al menos un símbolo 1.

Sean los siguientes estados:

- q_0: Estado inicial.
- q_{iNV} para $i \in \{1, \ldots, 4\}$: Si u es la cadena leída, tenemos que $|u| = i$ pero $n_1(u) = 0$, por lo que no es válido.
- q_{jI} para $j \in \{1, \ldots, 4\}$: Si u es la cadena leída, tenemos que $|u| = j$ y $n_1(u)$ es impar. Son estados finales.
- q_I: Si u es la cadena leída, tenemos que $|u| \geqslant 5$ y $n_1(u)$ es impar.

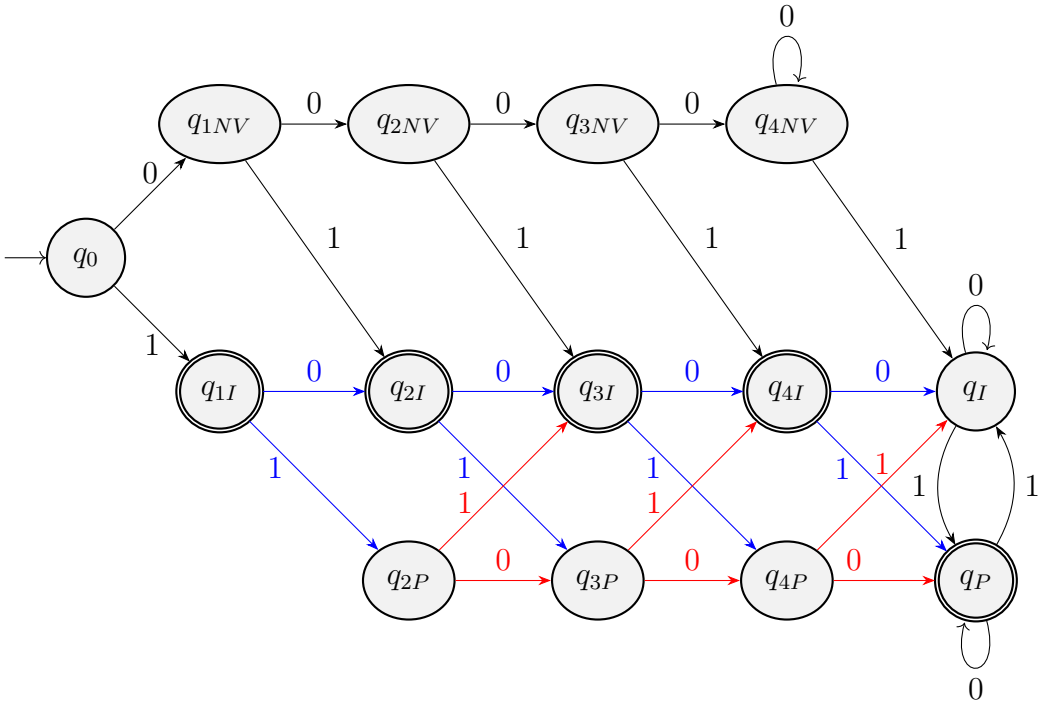


Figura 1.71: AFD asociado a L_1 del Ejercicio 1.3.23.

- q_{jP} para $j \in \{1, \ldots, 4\}$: Si u es la cadena leída, tenemos que $|u| = j$ y $n_1(u)$ es par. Además, $n_1(u) > 0$.
- q_P: Si u es la cadena leída, tenemos que $|u| \geqslant 5$ y $n_1(u)$ es par. Además, $n_1(u) > 0$. Es un estado final.

El AFD asociado a L_1 es el de la Figura 1.71.

Si el AFD de la Figura 1.71 es $M = (Q, \{0, 1\}, \delta, q_0, F)$, entonces la gramática pedida es $G = (Q, \{0, 1\}, P, q_0)$, donde P es:

$$P = \begin{cases} q_0 \to 0q_{1NV} \mid 1q_{1I}, \\ q_{1NV} \to 0q_{2NV} \mid 1q_{2I}, \\ q_{2NV} \to 0q_{3NV} \mid 1q_{3I}, \\ q_{3NV} \to 0q_{4NV} \mid 1q_{4I}, \\ q_{4NV} \to 0q_{4NV} \mid 1q_I, \\ q_I \to 0q_I \mid 1q_P, \\ q_P \to 0q_P \mid 1q_I \mid \varepsilon, \\ q_{1I} \to 0q_{2I} \mid 1q_{2P} \mid \varepsilon, \\ q_{2I} \to 0q_{3I} \mid 1q_{3P} \mid \varepsilon, \\ q_{3I} \to 0q_{4I} \mid 1q_{4P} \mid \varepsilon, \\ q_{4I} \to 0q_I \mid 1q_P \mid \varepsilon, \\ q_{2P} \to 0q_{3P} \mid 1q_{3I}, \\ q_{3P} \to 0q_{4P} \mid 1q_{4I}, \\ q_{4P} \to 0q_P \mid 1q_I \end{cases}$$

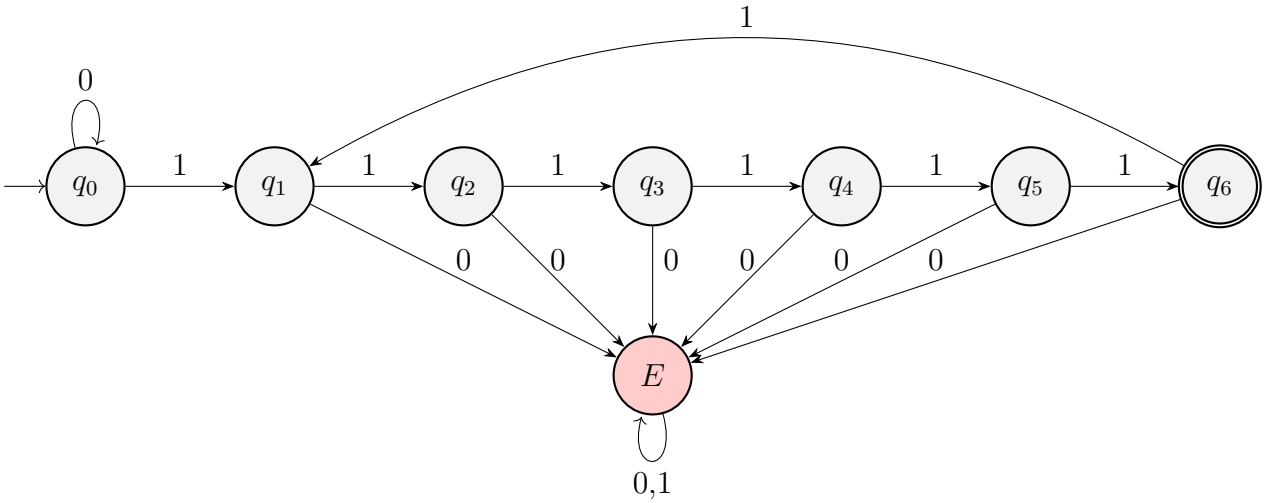


Figura 1.72: AFD minimal asociado a L_2 del Ejercicio 1.3.23.2.

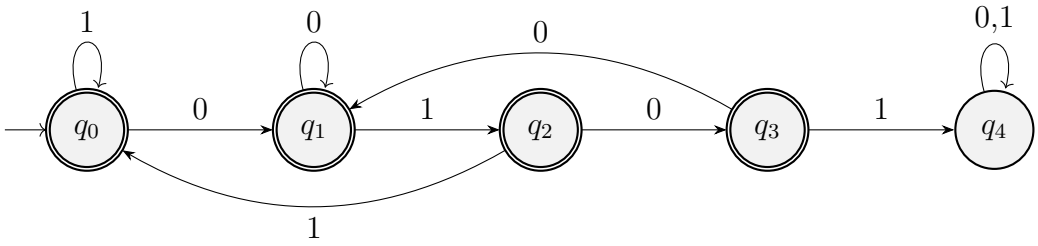


Figura 1.73: AFD minimal asociado a L_1 del Ejercicio 1.3.24.

2. Construye un autómata que reconozca el lenguaje L_2 dado por:

$$L_2 = \{0^n 1^m \mid n \geqslant 0, m \geqslant 1, m \text{ es múltiplo de } 6\}$$

   El autómata finito determinista asociado a L_2 es el de la Figura 1.72.

3. Diseña el AFD mínimo que reconozca el lenguaje $(L_1 \cup L_2)$.

   Razonando, llegamos a que $L_2 \subset L_1$. Por tanto, $L_1 \cup L_2 = L_1$, por lo que el AFD minimal asociado a $L_1 \cup L_2$ es el de la Figura 1.71.

Ejercicio 1.3.24. Encuentra para cada uno de los siguientes lenguajes una gramática de tipo 3 que lo genere o un autómata finito que lo reconozca:

- $L_1 = \{u \in \{0,1\}^* \mid u \text{ no contiene la subcadena “0101”}\}$.
- $L_2 = \{0^i 1^j 0^k \mid i \geqslant 1, k \geqslant 0, i \text{ impar }, k \text{ múltiplo de 3 y } j \geqslant 2\}$.

Diseña el AFD mínimo que reconoce el lenguaje $(L_1 \cap L_2)$.

El AFD minimal asociado a L_1 es el de la Figura 1.73.
El AFD minimal asociado a L_2 es el de la Figura 1.74.
El AFD asociado a $(L_1 \cap L_2)$ es el de la Figura 1.75. Notemos que todos los estados de la forma q_i, E se han agrupado en un único estado E, ya que todos ellos son indistingubles puesto que no se puede llegar desde ellos a ningún final.

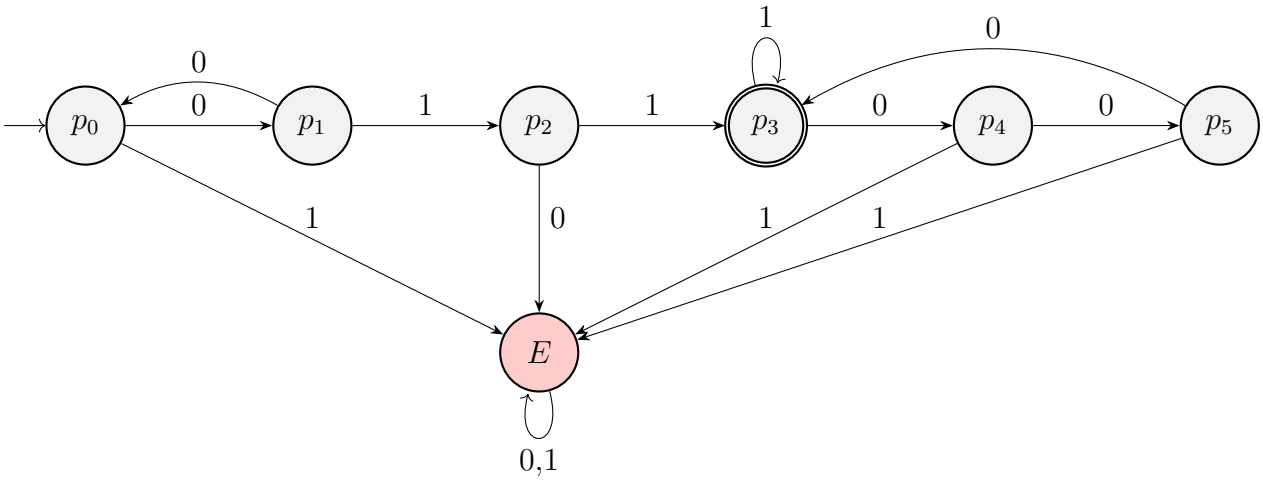


Figura 1.74: AFD minimal asociado a L_2 del Ejercicio 1.3.24.

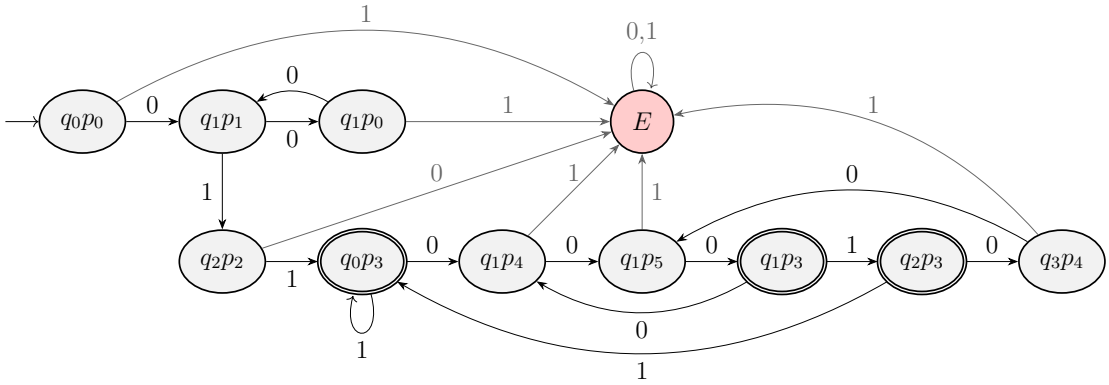


Figura 1.75: AFD asociado a $(L_1 \cap L_2)$ del Ejercicio 1.3.24.

	q_0p_0	q_1p_1	q_1p_0	q_2p_2	q_0p_3	q_1p_4	q_1p_5	q_1p_3	q_2p_3	q_3p_4
q_1p_1	×									
q_1p_0		×								
q_2p_2	×	×	×							
q_0p_3	×	×	×	×						
q_1p_4	×	×	×	×	×					
q_1p_5	×	~~(q_0p_0, q_3p_4)~~ ~~(q_1p_0, q_3p_4)~~	×	×	×	×				
q_1p_3	×	×	×	×		×	×			
q_2p_3	×	×	×	×	(q_1p_3, q_0p_3)	×	×	(q_1p_3, q_2p_3)		
q_3p_4	×	×	×	×	×	(q_0p_3, q_2p_3) (q_1p_3, q_2p_3)	×	×	×	
E	×	×	×	×	×	×	×	×	×	×

Tabla 1.14: Tabla de minimización del autómata de la Figura 1.75.

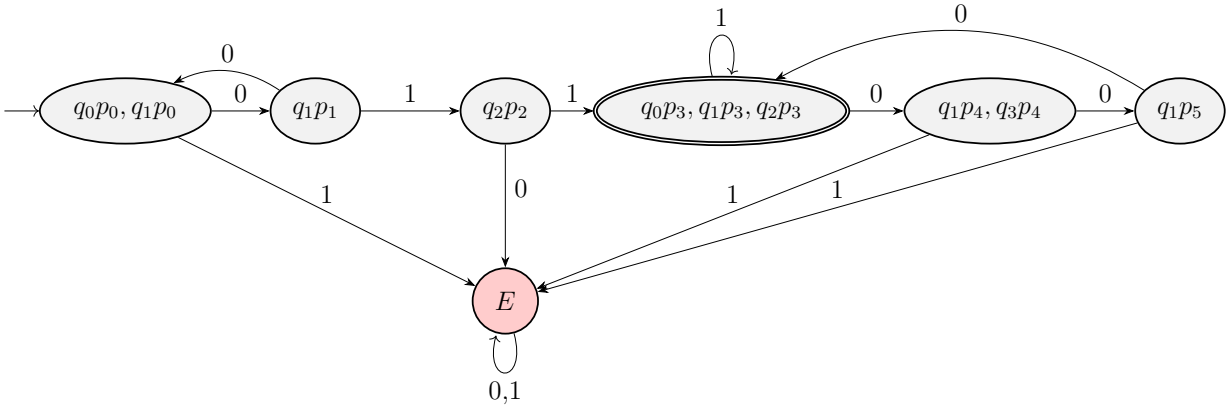


Figura 1.76: AFD minimal asociado a $(L_1 \cap L_2)$ del Ejercicio 1.3.24.

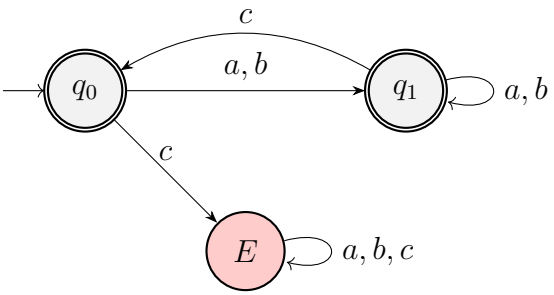


Figura 1.77: AFD minimal asociado al lenguaje del Ejercicio 1.3.25.1.

La minimización de este autómata se muestra en la Tabla 1.14.

El AFD minimal asociado a $(L_1 \cap L_2)$ es el de la Figura 1.76. Como podemos ver en el AFD minimal de la Figura 1.76, este es isomorfo al de la Figura 1.74, por lo que $L_1 \cap L_2 = L_2$. Esto es algo que podríamos haber deducido al principio, ya que si $L_2 \subset L_1$.

Ejercicio 1.3.25. Dado el alfabeto $A = \{a, b, c\}$, encuentra:

1. Un AFD que reconozca las palabras en las que cada "c" va precedida de una "a" o una "b".

2. Una expresión regular que represente el lenguaje compuesto por las palabras de longitud impar en las que el símbolo central es una "c".

   Veamos que este lenguaje no es regular usando el lema de bombeo. Para todo $n \in \mathbb{N}$, tomamos la palabra $z = a^n c a^n \in L$, con $|z| = 2n + 1 \geqslant n$. Toda descomposición de z en uvw, con $u, v, w \in \{0, 1\}^*$, $|uv| \leqslant n$ y $|v| \geqslant 1$ debe tener:

   $$u = a^k, \quad v = a^l, \quad w = a^{n-k-l} c a^n \qquad \text{con } l, k \in \mathbb{N} \cup \{0\}, l \geqslant 1, k + l \leqslant n$$

   Para $i = 2$, tenemos que $uv^2w = a^{k+2l+n-k-l} c a^n = a^{n+l} c a^n \notin L$ ya que, como $l \geqslant 1$, $n + l \neq n$, por lo que c no es el símbolo central. Por lo tanto, por el contrarrecíproco del lema de bombeo, L no es regular; y por tanto no podemos encontrar una expresión regular que lo represente.

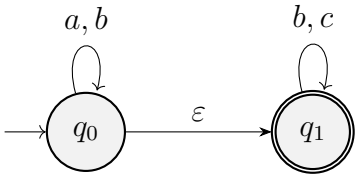


Figura 1.78: AFND asociado a L_1 del Ejercicio 1.3.26.

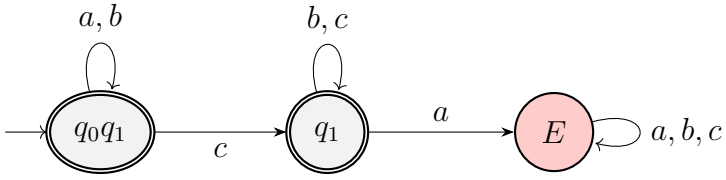


Figura 1.79: AFD minimal asociado a L_1 del Ejercicio 1.3.26.

3. Una gramática regular que genere las palabras de longitud impar.
   Sea $G = (\{S, X\}, A, P, S)$, donde P es:

$$S \to aX \mid bX \mid cX,$$
$$X \to aS \mid bS \mid cS \mid \varepsilon.$$

   Tenemos que $\mathcal{L}(G) = \{w \in A^* \mid |w| \text{ es impar}\}$.

Ejercicio 1.3.26. Construir autómatas finitos para los siguientes lenguajes sobre el alfabeto $\{a, b, c\}$:

1. L_1: palabras del lenguaje $(a + b)^*(b + c)^*$.
   El AFND asociado a L_1 es el de la Figura 1.78.
   El AFD minimal asociado a L_1 es el de la Figura 1.79.
2. L_2: palabras en las que nunca hay una "a" posterior a una "c".
   Vemos que L_2 tiene el mismo AFD que el de la Figura 1.79, por lo que $L_1 = L_2$.
3. $(L_1 \setminus L_2) \cup (L_2 \setminus L_1)$
   Como $L_1 = L_2$, tenemos que $(L_1 \setminus L_2) \cup (L_2 \setminus L_1) = \emptyset$.

Ejercicio 1.3.27. Si $f : \{0, 1\}^* \to \{a, b, c\}^*$ es un homomorfismo dado por

$$f(0) = aab \qquad f(1) = bbc$$

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.

El autómata finito minimal asociado a L es el de la Figura 1.80, donde q_i representa el estado en el que $n_a \mod 4 = i$. Empleando un distino número de cadenas de a's, vemos que los estados son distinguibles, por lo que es minimal.

Empleando el algoritmo, el AFD asociado a $f^{-1}(L)$ es el de la Figura 1.81. No obstante, y debido a que hay estados inaccesibles, este no es minimal. El AFD minimal asociado a $f^{-1}(L)$ es el de la Figura 1.82.

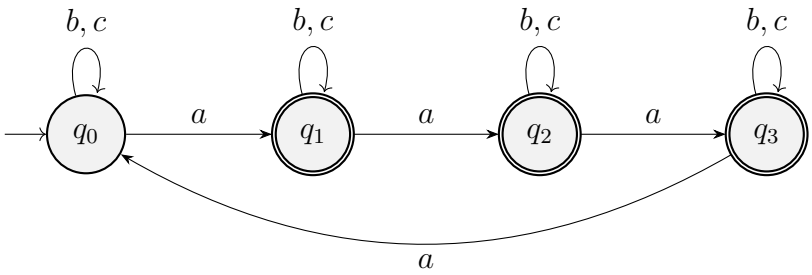


Figura 1.80: AFD minimal asociado a L del Ejercicio 1.3.27.

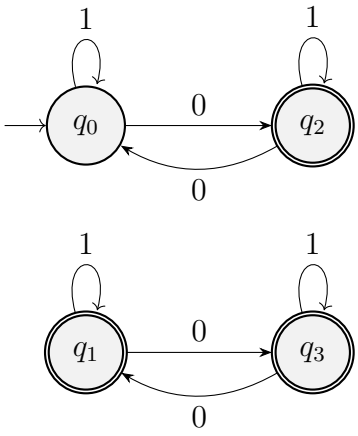


Figura 1.81: AFD asociado a $f^{-1}(L)$ del Ejercicio 1.3.27.

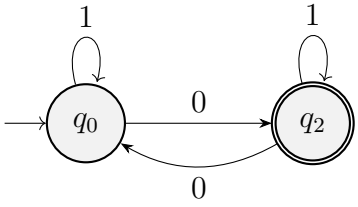


Figura 1.82: AFD minimal asociado a $f^{-1}(L)$ del Ejercicio 1.3.27.

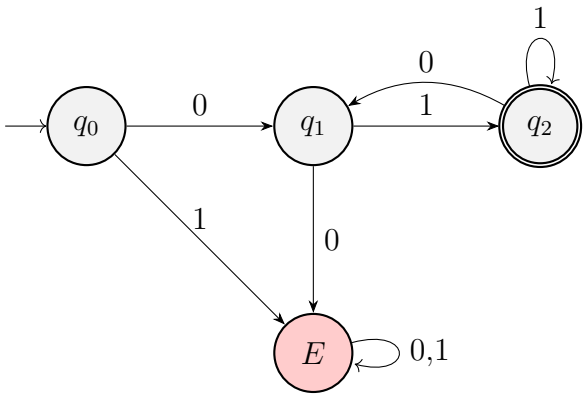


Figura 1.83: AFD minimal asociado a L_1 del Ejercicio 1.3.28.

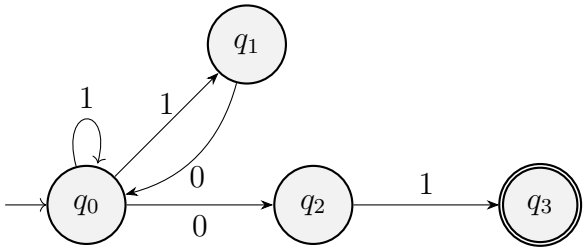


Figura 1.84: AFND asociado a L_2 del Ejercicio 1.3.28.

Ejercicio 1.3.28. 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 $L_1 \setminus L_2$.

El AFD minimal asociado a L_1 es el de la Figura 1.83.
El AFND asociado a L_2 es el de la Figura 1.84.
El AFD asociado a L_2 es el de la Figura 1.85.
El autómata asociado a $L_1 \setminus L_2$ es el de la Figura 1.86, donde hemos agrupado los estados de la forma E, q_i en E (ya que todos esos son indistinguibles puesto que no son finales y no se puede llegar a ellos desde un estado final).

Ejercicio 1.3.29. 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

$$f(a) = 01 \qquad f(b) = 1 \qquad f(c) = 0 \qquad f(d) = 00$$

El AFND asociado a L es el de la Figura 1.87.
El AFD minimal asociado a L es el de la Figura 1.88.
El AFD asociado a $f^{-1}(L)$ es el de la Figura 1.89.
Para obtener la expresión regular asociada a $f^{-1}(L)$, planteamos el sistema de ecuaciones siguiente:

$$\begin{cases} q_0 = bq_0 + cq_0q_1 + aq_0q_2 + dq_0q_1q_2 \\ q_0q_1 = (c+d)q_0q_1q_2 + (a+b)q_0q_2 \\ q_0q_1q_2 = (c+d)q_0q_1q_2 + (a+b)q_0q_2 + \varepsilon \\ q_0q_2 = cq_0q_1 + bq_0 + aq_0q_2 + dq_0q_1q_2 + \varepsilon \end{cases}$$

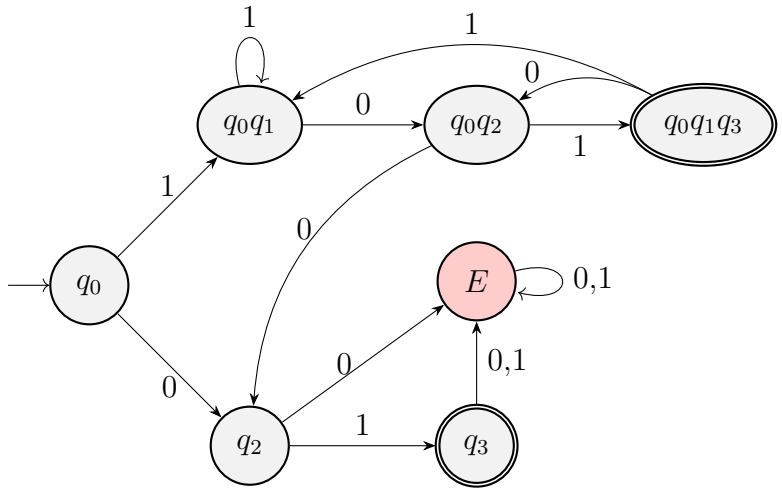


Figura 1.85: AFD minimal asociado a L_2 del Ejercicio 1.3.28.

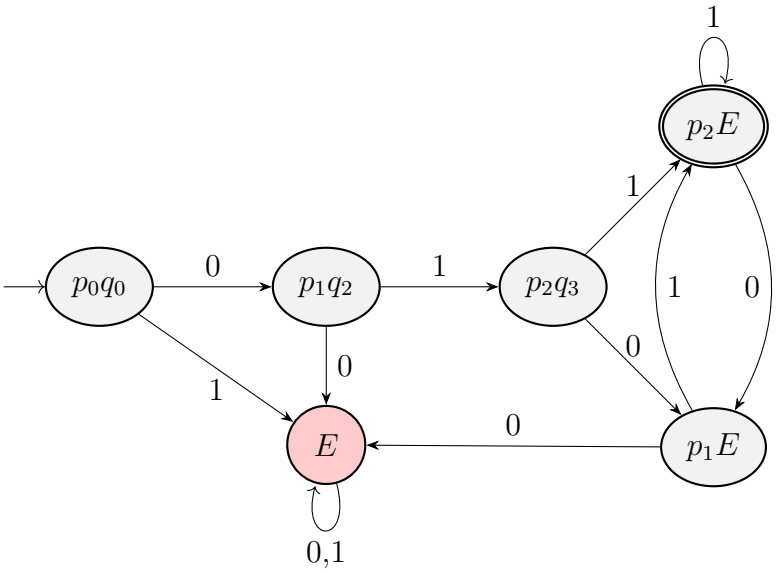


Figura 1.86: AFD minimal asociado a $L_1 \setminus L_2$ del Ejercicio 1.3.28.

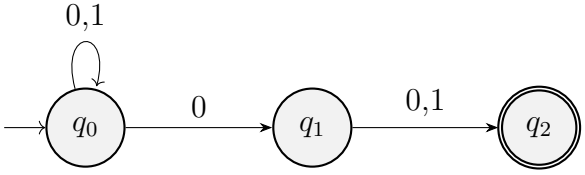


Figura 1.87: AFND asociado a L del Ejercicio 1.3.29.

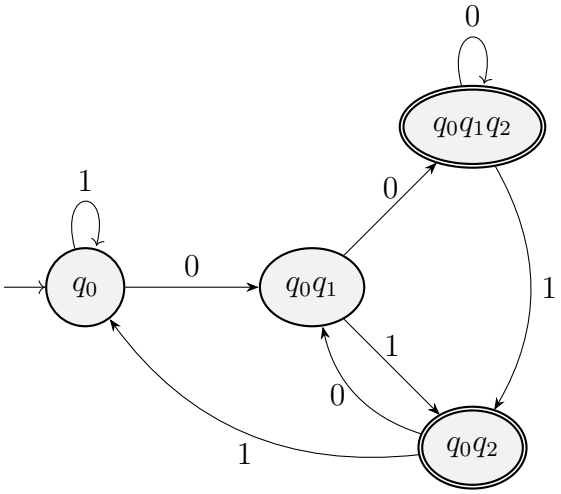


Figura 1.88: AFD minimal asociado a L del Ejercicio 1.3.29.

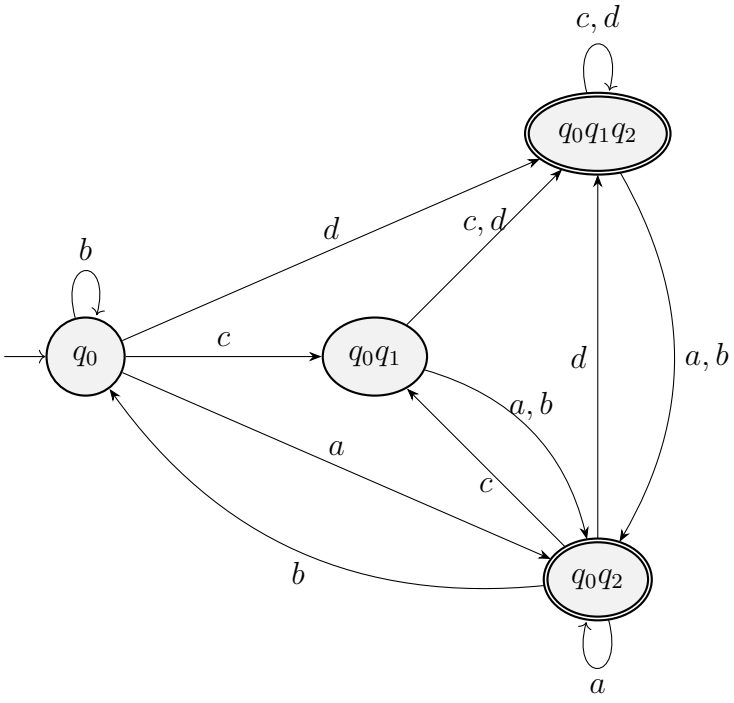


Figura 1.89: AFD asociado a $f^{-1}(L)$ del Ejercicio 1.3.29.

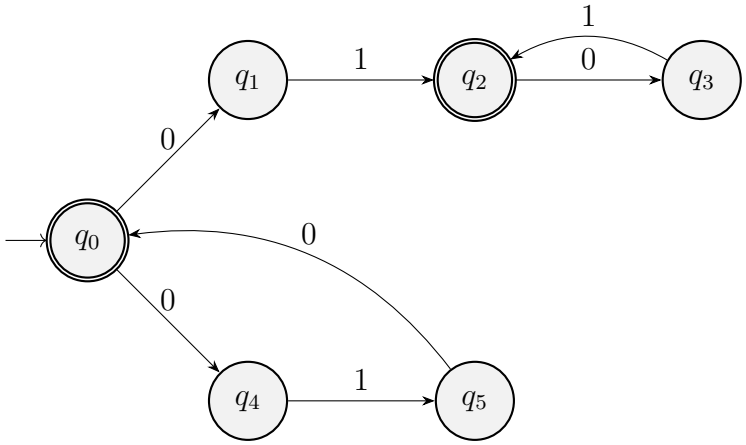


Figura 1.90: AFND asociado a la expresión regular del Ejercicio 1.3.30.

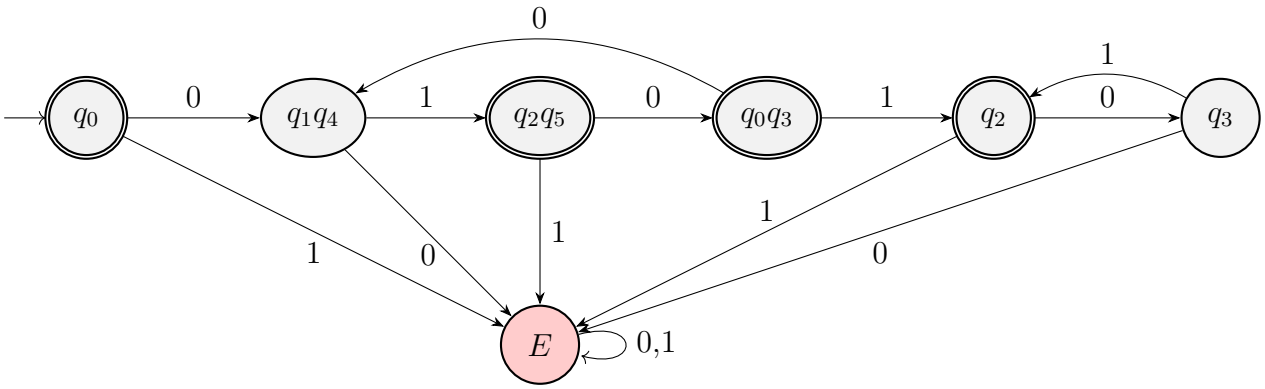


Figura 1.91: AFD asociado a la expresión regular del Ejercicio 1.3.30.

Sería necesario resolver el sistema para obtener la expresión regular.

Ejercicio 1.3.30. Obtener un autómata finito determinista para el lenguaje asociado a la expresión regular: $(01)^+ + (010)^*$. Minimizarlo.

El AFND asociado a la expresión regular es el de la Figura 1.90.
El AFD asociado a la expresión regular es el de la Figura 1.91.
La minimización del autómata se encuentra en la Tabla 1.15.
Por tanto, como todos los estados del AFD de la Figura 1.91 son distinguibles, este es minimal.

q_1q_4	×					
q_2q_5	×	×				
q_0q_3	×	×	×			
q_2	×	×	~~(q_3, q_1q_4)~~	×		
q_3	×	~~(q_0, q_2)~~	×	×	×	
E	×	×	×	×	×	×
	q_0	q_1q_4	q_2q_5	q_0q_3	q_2	q_3

Tabla 1.15: Minimización del autómata del Ejercicio 1.3.30.

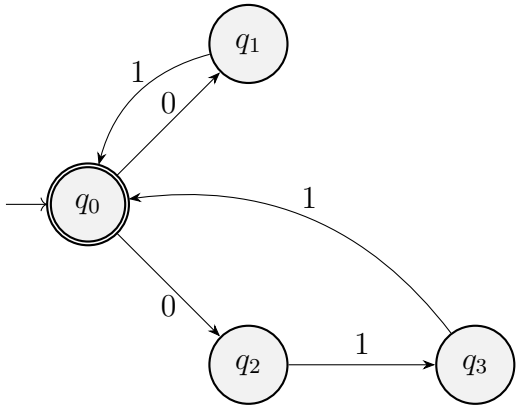


Figura 1.92: AFND asociado al lenguaje L del Ejercicio 1.3.31.

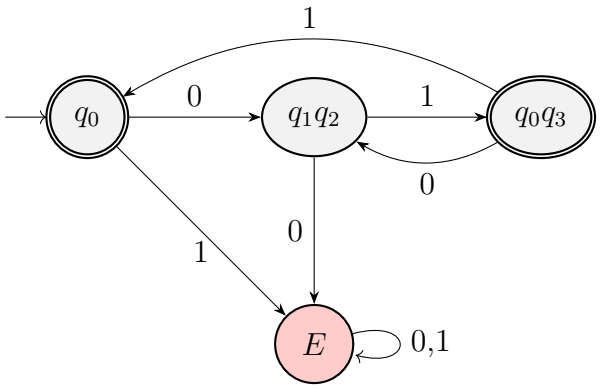


Figura 1.93: AFD asociado al lenguaje L del Ejercicio 1.3.31.

Ejercicio 1.3.31. Dado el lenguaje L asociado a la expresión regular $(01 + 011)^*$ y el homomorfismo $f : \{0,1\}^* \to \{0,1\}^*$ dado por $f(0) = 01$, $f(1) = 1$, construir una expresión regular para el lenguaje $f^{-1}(L)$.

El AFND asociado a la expresión regular es el de la Figura 1.92.

El AFD asociado a la expresión regular es el de la Figura 1.93.

El AFD asociado a $f^{-1}(L)$ es el de la Figura 1.94.

No obstante, este no es minimal, puesto que tiene estados inaccesibles. El AFD minimal asociado a $f^{-1}(L)$ es el de la Figura 1.95.

Para obtener la expresión regular asociada a $f^{-1}(L)$, planteamos el sistema de ecuaciones siguiente:

$$\begin{cases} q_0 = 0q_0q_3 + 1E + \varepsilon \\ q_0q_3 = 0q_0q_3 + 1q_0 + \varepsilon \\ E = (0+1)E \end{cases}$$

Por el Lema de Arden, tenemos que $E = (0+1)^*\emptyset = \emptyset$ y $q_0q_3 = 0^*(1q_0 + \varepsilon)$. Por tanto, tenemos que:

$$\begin{aligned} q_0 &= 00^*(1q_0 + \varepsilon) + \varepsilon = 0^+(1q_0 + \varepsilon) + \varepsilon \Longrightarrow \\ \Longrightarrow q_0 &= (0^+1)^*(0^+ + \varepsilon) \end{aligned}$$

Por tanto, la expresión regular asociada a $f^{-1}(L)$ es $(0^+1)^*(0^+ + \varepsilon)$.

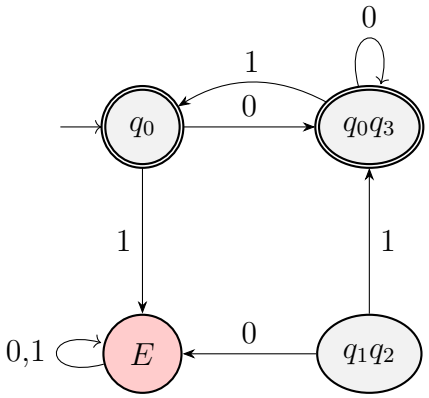


Figura 1.94: AFD asociado a $f^{-1}(L)$ del Ejercicio 1.3.31.

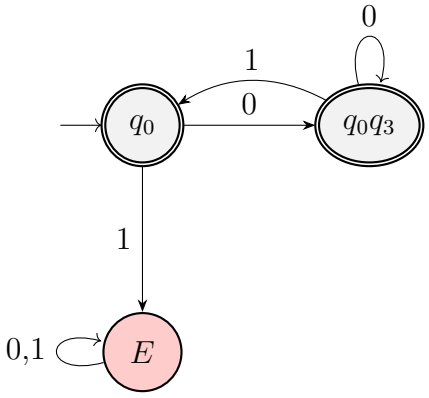


Figura 1.95: AFD minimal asociado a $f^{-1}(L)$ del Ejercicio 1.3.31.

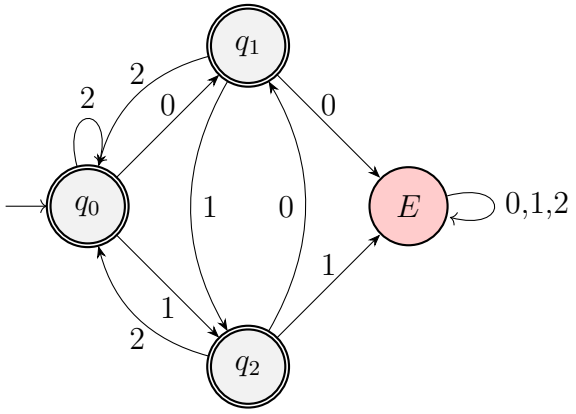


Figura 1.96: AFD minimal asociado a L del Ejercicio 1.3.32.

Ejercicio 1.3.32. 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.

   El AFD minimal asociado a L es el de la Figura 1.96.

   Para obtener la expresión regular asociada a L, planteamos el sistema de ecuaciones siguiente:

   $$\begin{cases} q_0 = 0q_1 + 1q_2 + 2q_0 + \varepsilon \\ q_1 = 1q_2 + 0E + 2q_0 + \varepsilon \\ q_2 = 1E + 0q_1 + 2q_0 + \varepsilon \\ E = (0 + 1 + 2)E \end{cases}$$

   Por el Lema de Arden, tenemos que $E = (0 + 1 + 2)^*\emptyset = \emptyset$, por lo que $q_2 = 0q_1 + 2q_0 + \varepsilon$. Por tanto:

   $$\begin{aligned} q_1 &= 1(0q_1 + 2q_0 + \varepsilon) + 0\emptyset + 2q_0 + \varepsilon = 1(0q_1 + 2q_0 + \varepsilon) + 2q_0 + \varepsilon \Longrightarrow \\ \Longrightarrow q_1 &= (10)^*(1 + (1 + \varepsilon)2q_0 + \varepsilon) = \\ &= (10)^*(1 + \varepsilon)(2q_0 + \varepsilon) \end{aligned}$$

   Por tanto, la expresión regular asociada a L es:

   $$\begin{aligned} q_0 &= 0(10)^*(1 + \varepsilon)(2q_0 + \varepsilon) + 1(0(10)^*(1 + \varepsilon)(2q_0 + \varepsilon) + 2q_0 + \varepsilon) + 2q_0 + \varepsilon = \\ &= 0(10)^*(1 + \varepsilon)(2q_0 + \varepsilon) + 10(10)^*(1 + \varepsilon)(2q_0 + \varepsilon) + 1(2q_0 + \varepsilon) + 2q_0 + \varepsilon = \\ &= [0(10)^*(1 + \varepsilon) + 10(10)^*(1 + \varepsilon) + 1 + \varepsilon](2q_0 + \varepsilon) = \\ &= [0(10)^* + 10(10)^* + \varepsilon](1 + \varepsilon)(2q_0 + \varepsilon) = \\ &= [(1 + \varepsilon)0(10)^* + \varepsilon](1 + \varepsilon)(2q_0 + \varepsilon) = \\ &= [[(1 + \varepsilon)0(10)^* + \varepsilon](1 + \varepsilon)2]^*[(1 + \varepsilon)0(10)^* + \varepsilon](1 + \varepsilon) \end{aligned}$$

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 $f(L)$.

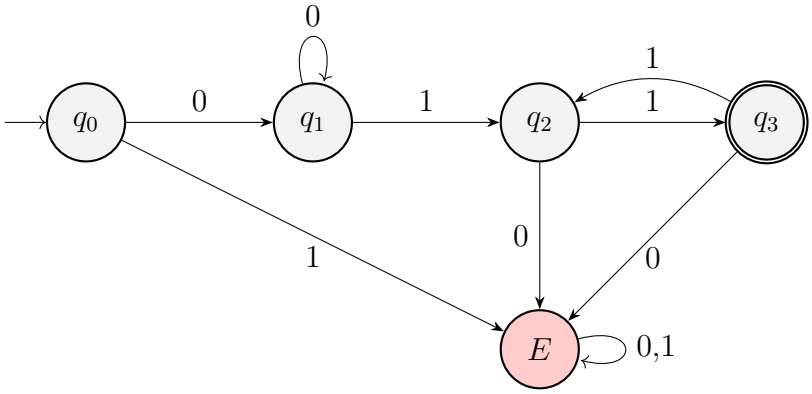


Figura 1.97: AFD minimal asociado a L_1 del Ejercicio 1.3.33.

La expresión regular asociada a $f(L)$ es:

$$[[(100+\varepsilon)001(100001)^*+\varepsilon](100+\varepsilon)0011]^*[(100+\varepsilon)001(100001)^*+\varepsilon](100+\varepsilon)$$

3. Dar una expresión regular para LL^{-1}.

   Razonando los AFD vemos que $L = L^{-1}$, por lo que la expresión regular asociada a LL^{-1} es:

$$[[[(1+\varepsilon)0(10)^*+\varepsilon](1+\varepsilon)2]^*[(1+\varepsilon)0(10)^*+\varepsilon](1+\varepsilon)]^2$$

Ejercicio 1.3.33. Dados los lenguajes

$$L_1 = \{0^i 1^j \mid i \geqslant 1, j \text{ es par y } j \geqslant 2\}$$
$$L_2 = \{1^j 0^k \mid k \geqslant 1, j \text{ es impar y } j \geqslant 1\}$$

Encuentre:

1. Una gramática regular que genere el lenguaje L_1.

   El AFD minimal asociado a L_1 es el de la Figura 1.97.

   La gramática regular asociada a L_1 por tanto que lo genera es $G = (V, \{0,1\}, P, q_0)$ donde P es:

$$V = \{q_0, q_1, q_2, q_3\}$$
$$P = \begin{cases} q_0 \to 0q_1 \\ q_1 \to 0q_1 \mid 1q_2 \\ q_2 \to 1q_3 \\ q_3 \to 1q_2 \mid \varepsilon \end{cases}$$

2. Una expresión regular que represente al lenguaje L_2.

   El AFD minimal asociado a L_2 es el de la Figura 1.98.

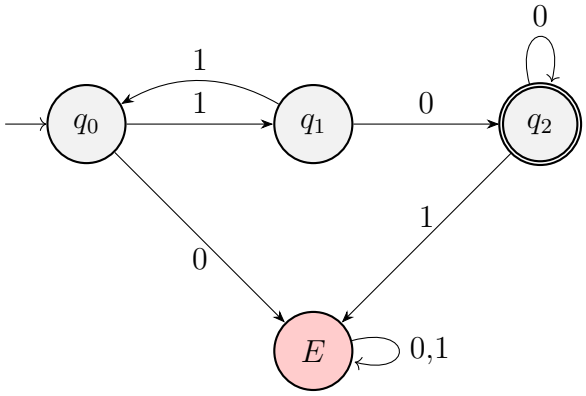


Figura 1.98: AFD minimal asociado a L_2 del Ejercicio 1.3.33.

δ_1	A	B	C	D	E	F	G	H
0	B	G	A	C	H	C	G	G
1	F	C	C	G	F	G	E	C

Tabla 1.16: Transiciones del autómata M_1 del Ejercicio 1.3.34.

La expresión regular asociada a L_2 se obtiene resolviendo el sistema de ecuaciones siguiente:

$$\begin{cases} q_0 = 1q_1 + 0E \\ q_1 = 1q_0 + 0q_2 \\ q_2 = 0q_2 + 1E + \varepsilon \\ E = (0+1)E \end{cases}$$

Por el Lema de Arden, tenemos que $E = (0+1)^*\emptyset = \emptyset$ y $q_2 = 0^*\varepsilon = 0^*$. Por tanto:

$$q_1 = 1q_0 + 00^+ = 1q_0 + 0^+$$

Por tanto, la expresión regular asociada a L_2 es:

$$q_0 = 1(1q_0 + 0^+) = (11)^*(10^+)$$

3. Un automata finito determinista que acepte las cadenas de la concatenación de los lenguajes L_1 y L_2. Aplica el algoritmo para minimizar este autómata.

   El AFD asociado a L_1L_2 es el de la Figura 1.99. Este es:

$$\begin{aligned} L_1L_2 &= \{0^i1^{j+j'}0^k \mid i \geqslant 1, j \text{ es par y } j \geqslant 2, j' \text{ es impar y } j' \geqslant 1, k \geqslant 1\} = \\ &= \{0^i1^j0^k \mid i \geqslant 1, j \text{ es impar y } j \geqslant 3, k \geqslant 1\} \end{aligned}$$

   Este autómata vemos de forma directa que es minimal, puesto que desde cada estado para llegar al único estado final hemos de leer una cadena distinta.

Ejercicio 1.3.34. Considerar los AFD $M_1 = (\{A, B, C, D, E, F, G, H\}, \{0, 1\}, \delta_1, A, \{C\})$ y $M_2 = (\{A', B', C', D', G'\}, \{0, 1\}, \delta_2, A', \{D'\})$ donde δ_1 y δ_2 están definidas por las Tablas 1.16 y 1.17 respectivamente. Determinar si ambos autómatas finitos generan

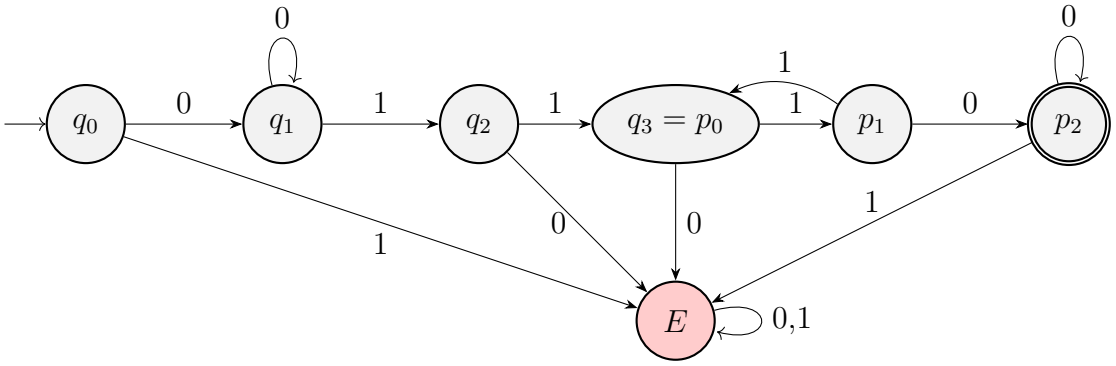


Figura 1.99: AFD asociado a L_1L_2 del Ejercicio 1.3.33.

δ_2	A'	B'	C'	D'	G'
0	G'	B'	D'	A'	B'
1	C'	A'	B'	D'	D'

Tabla 1.17: Transiciones del autómata M_2 del Ejercicio 1.3.34.

el mismo lenguaje.

Como M_1 tiene más estados, comenzamos minimizando este autómata. Veamos cuáles son sus estados accesibles:

$$\{A, B, F, G, C, E, H\} = Q \setminus \{D\}$$

Por tanto, tenemos que el único estado no accesible es D. La tabla de minimalización se encuentra en la Tabla 1.18.

Por tanto, notando por $\equiv$ a la relación de indistinguibilidad, tenemos que:

$$H \equiv B \qquad A \equiv E$$

Por tanto, el autómata minimal de M_1 es:

$$M_1^{\min} = \{\{(AE), (BH), C, F, G\}, \{0, 1\}, \delta_1^{\min}, (AE), \{C\}\}$$

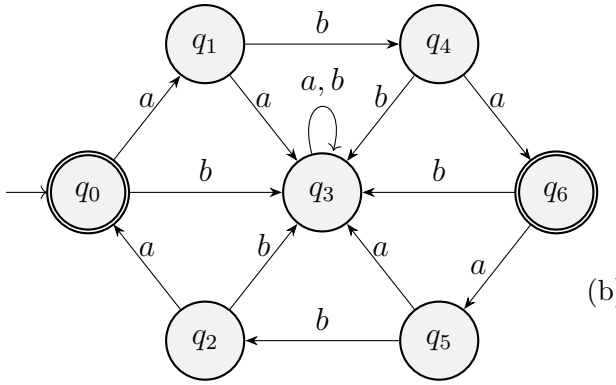
donde $\delta_1^{\min}$ está definida por la Tabla 1.19.

B	×					
C	×	×				
E		×	×			
F	×	×	×	×		
G	×	×	×	×	×	
H	×	(E, A)	×	×	×	×
	A	B	C	E	F	G

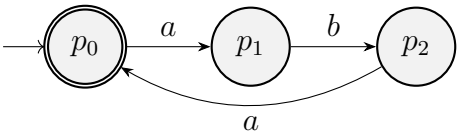
Tabla 1.18: Minimalización del autómata M_1 del Ejercicio 1.3.34.

$\delta_1^{\min}$	(AE)	(BH)	C	F	G
0	(BH)	G	(AE)	C	G
1	F	C	C	G	(AE)

Tabla 1.19: Transiciones del autómata $M_1^{\min}$ del Ejercicio 1.3.34.



(a) Autómata M_1 del Ejercicio 1.3.35.



(b) Autómata M_2 del Ejercicio 1.3.35.

Figura 1.100: Autómatas del Ejercicio 1.3.35.

Como podemos ver, $M_1^{\min}$ y M_2 son isomorfos, con isomorfismo f dado por:

$$\begin{aligned} f((AE)) &= A' \qquad \text{(inicial)} \\ f(C) &= D' \qquad \text{(final)} \\ f((BH)) &= G' \\ f(F) &= C' \\ f(G) &= B' \end{aligned}$$

Por tanto, tenemos que $\mathcal{L}(M_1) = \mathcal{L}(M_2)$.

Ejercicio 1.3.35. Comprobar si los autómatas de las Figuras 1.100a y 1.100b generan el mismo lenguaje.

En primer lugar, vemos que q_3 es distinguible del resto, pues que es el único estado desde el cual no se puede llegar a un estado final. La tabla de minimalización de M_1 se encuentra en la Tabla 1.20.

Por tanto, el autómata minimal de M_1 es el de la Figura 1.101.

Como vemos, el autómata minimal de M_1 es isomorfo al AFD asociado a M_2 (ya que cuenta con un estado de error con las transiciones restantes) con isomorfismo f

q_1	×					
q_2	×	×				
q_3	×	×	×			
q_4	×	×	(q_1, q_5)	×		
q_5	×	(q_0, q_6)	×	×	×	
q_6	(q_2, q_4)	×	×	×	×	×
	q_0	q_1	q_2	q_3	q_4	q_5

Tabla 1.20: Minimalización del autómata M_1 del Ejercicio 1.3.35.

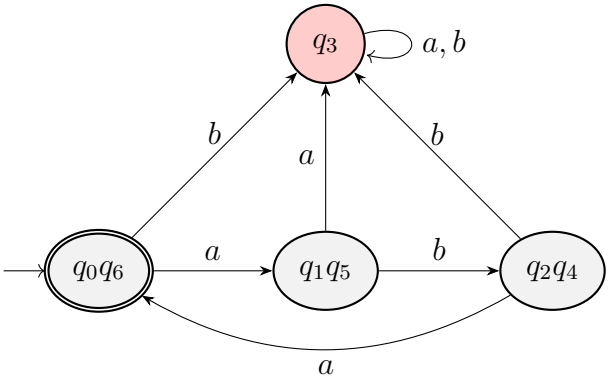


Figura 1.101: Autómata minimal asociado a M_1 del Ejercicio 1.3.35.

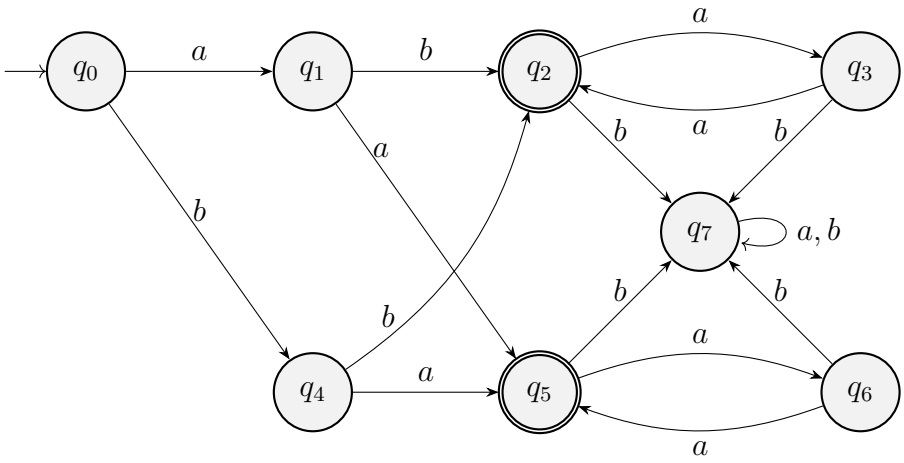


Figura 1.102: Autómata a minimizar del Ejercicio 1.3.36.

dado por:

$$\begin{aligned} f(q_0q_6) &= p_0 \qquad \text{(inicial)} \\ f(q_1q_5) &= p_1 \\ f(q_2q_4) &= p_2 \qquad \text{(final)} \end{aligned}$$

Por tanto, tenemos que $\mathcal{L}(M_1) = \mathcal{L}(M_2)$.

Ejercicio 1.3.36. Minimizar el autómata de la Figura 1.102.

En primer lugar, vemos que q_7 es distinguible del resto, pues que es el único estado desde el cual no se puede llegar a un estado final. La tabla de minimalización de M se encuentra en la Tabla 1.21.

Por tanto, si notamos por $\equiv$ a la relación de indistinguibilidad, tenemos que:

$$q_4 \equiv q_1 \qquad q_5 \equiv q_2 \qquad q_6 \equiv q_3$$

Por tanto, el autómata minimal de M es el de la Figura 1.103.

q_1	×						
q_2	×	×					
q_3	×	×	×				
q_4	×		×	×			
q_5	×	×	(q_3, q_6)	×	×		
q_6	×	×	×	(q_2, q_5)	×	×	
q_7	×	×	×	×	×	×	×
	q_0	q_1	q_2	q_3	q_4	q_5	q_6

Tabla 1.21: Minimalización del autómata M del Ejercicio 1.3.36.

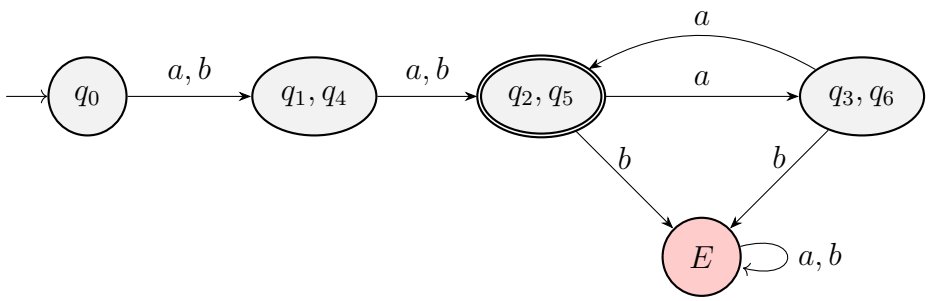


Figura 1.103: Autómata minimal asociado a M del Ejercicio 1.3.36.

Ejercicio 1.3.37. Si L_1 y L_2 son lenguajes sobre el alfabeto A, entonces *la mezcla perfecta* de estos lenguajes se define como el lenguaje:

$$\{w \mid w = a_1 b_1 \cdots a_k b_k \mid a_1 \dots a_k \in L_1, b_1 \dots b_k \in L_2, a_i, b_i \in A\}$$

Demostrar que si L_1 y L_2 son regulares, entonces la mezcla perfecta de L_1 y L_2 es regular.

Vamos a definir un autómata finito determinista que acepte la mezcla perfecta de L_1 y L_2. Sean los autómatas $M_1 = (Q_1, A, \delta_1, q_0^1, F_1)$ el autómata asociado a L_1 y $M_2 = (Q_2, A, \delta_2, q_0^2, F_2)$ el autómata asociado a L_2. Definimos el autómata $M = (Q, A, \delta, q_0, F)$ como sigue:

$$\begin{aligned} Q &= Q_1 \times Q_2 \times \{\mathcal{L}_1, \mathcal{L}_2\} \\ F &= F_1 \times F_2 \times \{\mathcal{L}_1\} \\ q_0 &= (q_0^1, q_0^2, \mathcal{L}_1) \\ \delta((q_1, q_2, \mathcal{L}_1), u) &= (\delta_1(q_1, u), q_2, \mathcal{L}_2) \qquad \forall u \in A \\ \delta((q_1, q_2, \mathcal{L}_2), u) &= (q_1, \delta_2(q_2, u), \mathcal{L}_1) \qquad \forall u \in A \end{aligned}$$

Notemos que el estado $(q_i, q_j, \mathcal{L}_k)$ indica que en el autómata M_1 se está en el estado q_i, en el autómata M_2 se está en el estado q_j y, ahora mismo, debemos leer una palabra de L_k. Por tanto, el autómata M acepta la mezcla perfecta de L_1 y L_2, por lo que es regular.

Ejercicio 1.3.38. Si L es un lenguaje, sea $L_{1/2}$ el conjunto de palabras que son las mitades de palabras de L y $L_{-1/3}$ el conjunto de palabras que son las dos terceras

partes de palabras de L. Es decir:

$$L_{1/2} = \{x \mid \exists y \in A^*, |x| = |y|, xy \in L\}$$
$$L_{-1/3} = \{xz \mid \exists y \in A^*, |x| = |y| = |z|, xyz \in L\}$$

Demostrar que si L es regular, entonces $L_{1/2}$ también lo es, pero que $L_{-1/3}$ no es necesariamente regular.

1.3.1. Preguntas Tipo Test

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

1. El lema de bombeo puede usarse para demostrar que un lenguaje determinado es regular.

   Falso, el lema de bombeo nos dice que si un lenguajes es regular, entonces este cumple una determinada propiedad. Podemos usar su contrarrecíproco para ver que si una palabra del lenguaje no cumple dicha propiedad entonces el lenguaje no es regular, pero no nos sirve para determinar si un lenguaje lo es o no.

2. Todo lenguaje con un número finito de palabras es regular.

   Verdadero, ya que si tenemos $L = \{v_1, v_2, \ldots, v_n\}$ un lenguaje finito de $n \in \mathbb{N}$ palabras, entonces podemos construir la gramática $G = (\{S\}, A, P, S)$ con A el alfabeto sobre el que está definido L y P el siguiente conjunto de producciones:

   $$P = \{S \to v_1 \mid v_2 \mid \ \ldots \ \mid v_n\}$$

   Con lo que G es una gramática regular de forma que $L = L(G)$, con lo que es regular.

3. La intersección de lenguajes regulares es siempre regular.

   Verdadero. Como hemos visto en teoría, si tenemos dos lenguajes regulares, entonces podemos construir un autómata finito determinista para cada uno de ellos y construir el autómata producto, que genera la intersección de ambos lenguajes, con lo que el lenguaje como resultado de intersecar los dos lenguajes es regular.

4. La demostración del lema de bombeo se basa en que si leemos una palabra de longitud mayor o igual al número de estados del autómata, entonces en el camino que se recorre en el diagrama de transición se produce un ciclo.

   Verdadero, si tenemos un autómata finito determinista de n estados que reconoce un lenguaje regular, si leemos una palabra de longitud m con $m \geqslant n$, entonces en el "camino de lectura" de la palabra, hemos de pasar por $m + 1$ estados, con lo que pasaremos por al menos un estado dos veces o más, con lo que el autómata tendrá un ciclo.

5. Es más fácil determinar si una palabra pertenece a un lenguaje regular cuando éste viene dado por una expresión regular que cuando viene dado por un autómata finito determinista.

   Falso, si tenemos un autómata finito determinista que acepta el lenguaje en cuestión, ver si la palabra está en el lenguaje será tan sencillo como ir realizando las transiciones entre estados en el autómata leyendo la palabra y comprobando si al final llegamos a un estado final o si no. Por otra parte, puede suceder que ver si una palabra está en un lenguaje o no a partir de la expresión regular puede que sea sencillo, pero si consideramos una expresión regular muy compleja, posiblemente no sea tan fácil determinar si la palabra está en el lenguaje o no. En cualquier caso, reconocer si una palabra está en el lenguaje por un autómata finito determinista es siempre el proceso más sencillo.

6. En la demostración de que todo autómata finito tiene una expresión regular que representa el mismo lenguaje, el conjunto R_{ij}^k se define como el lenguaje de todas las palabras que llevan al autómata del estado q_i al estado q_j pasando por el estado número k, q_k.

   Falso, (se trata de una pregunta del Tema 2) ya que R_{ij}^k se define como el conjunto de todas las palabras que llevan al autómata del estado q_i al estado q_j pasando por estados de numeración menor o igual que k, pero no necesariamente por q_k.

7. El conjunto de todas las expresiones regulares es un lenguaje regular.

   Falso, si consideramos que la buena parentización[1] forma parte de las expresiones regulares, podemos demostrar que el conjunto de todas las expresiones regulares no es un lenguaje regular usando el Lema de bombeo:

   Sea $n \in \mathbb{N}$ y $a \in A$, consideramos[2] $z = ({}^n a[+a)]^n$. Es decir:

   $$z = (((\ldots(((a + a) + a) + a)\ldots + a) + a) + a)$$

   donde hay n paréntesis de apertura y n paréntesis de cierre y notemos que $|z| \geqslant n$. Supongamos que hay $u, v, w \in A^*$ tales que $z = uvw$ con $|v| \geqslant 1$ y $|uv| \leqslant n$. Entonces, $u = ({}^k$, $v = ({}^h$ con $h \geqslant 1$, $k + h \leqslant n$ y:

   $$w = ({}^{n-k-h}a + a) + a)\ldots + a) + a) + a) = ({}^{n-k-h}a[+a)]^n$$

   Sin embargo, $uv^0w = uw = ({}^{n-h}a[+a)]^n$, que no es una expresión regular correcta por no estar bien parentizada ($k \geqslant 1$), con lo que dicho lenguaje no es regular.

8. A partir de la demostración de que si R es regular y L un lenguaje cualquiera, entonces R/L es regular, se puede obtener un algoritmo para construir el

[1] El poner paréntesis.

[2] Donde hemos usado corchetes para diferenciarlos de los paréntesis del lenguaje.

autómata asociado a R/L.

Falso en general: si el lenguaje L es finito o regular, sí que nos dice cómo podemos construir el autómata asociado a R/L, pero si no puede suceder que no seamos capaces de hacerlo, por tener L infinitas palabras y no poder considerar en el autómata todas ellas.

9. En un autómata finito no-determinista, si intercambio entre sí los estados finales y no finales obtengo un autómata que acepta el lenguaje complementario.

Falso, ya que un autómata finito no determinista puede que no tenga definidas alguna transición desde un estado leyendo algún caracter (lo que en el determinista asociado significaría ir a un estado de error), con lo que al realizar dicho cambio en el autómata no determinista, no consideramos dichas transiciones. Para ver esto más claro, consideramos el autómata finito no determinista de la Figura 1.104, que acepta el lenguaje:

$$L = \{01u \mid u \in \{0,1\}^*\}$$

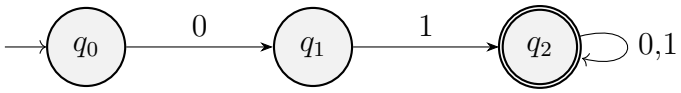


Figura 1.104: Autómata finito no determinista para la pregunta 9.

Si realizamos el cambio mencionado en la pregunta, entonces consideramos ahora como conjunto de estados finales $F' = \{q_0, q_1\}$, con lo que según esta pregunta, el nuevo autómata debería reconocer el lenguaje:

$$\overline{L} = \{w \mid w \neq 01u \quad \forall u \in \{0,1\}^*\}$$

Sin embargo, $110 \in \overline{L}$ y si intentamos leer esta palabra en el nuevo autómata finito no determinista, no la reconoce, como resultado de la falta de transiciones en el autómata de la Figura 1.104, como habíamos enunciado anteriormente.

10. Si en un autómata finito no hay estados distinguibles de nivel 2, ya no puede haber estados distinguibles de nivel 4.

11. Todo lenguaje generado por una gramática lineal por la derecha es también generado por una gramática lineal por la izquierda.

Verdadero, se vió en el Tema 2.

12. Un autómata finito determinista sin estados inaccesibles ni indistinguibles es minimal.

Verdadero, gracias a un resultado visto en teoría.

13. Si L es una lenguaje sobre el alfabeto A, entonces CAB(L) es siempre igual al cociente L/A^*.

    Verdadero, por definición de CAB(L) y de L/A^*:

    $$L/A^* = \{u \in A^* \mid \exists v \in A^* \text{ verificando } uv \in L\} = \text{CAB}(L)$$

14. El lenguaje de las palabras sobre $\{0, 1\}$ en las que la diferencia entre el número de ceros y unos es impar es regular.

    Verdadero, ya que se puede reconocer por el autómata de la Figura 1.105.

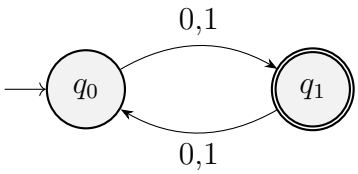


    Figura 1.105: Autómata finito determinista para la pregunta 14.

    Donde usamos el estado q_0 para representar que la diferencia entre el número de ceros y unos es par y q_1 para representar que la diferencia entre el número de ceros y unos es impar.

15. En un autómata finito cualquiera, si las transiciones dan lugar a un ciclo, entonces el lenguaje aceptado es infinito.

    Falso, solo será cierto si tras salir de dicho ciclo se puede llegar a un estado final. Para ilustrar este caso, observamos el autómata finito determinista de la Figura 1.106.

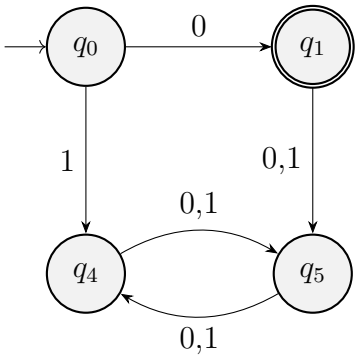


    Figura 1.106: Autómata finito determinista para la pregunta 15.

    En el autómata hay presente un ciclo y este reconoce el lenguaje $\{0\}$.

16. La expresión recursiva que se emplea para obtener la expresión regular asociada a un autómata finito determinista es: $r_{ij}^k = r_{ij}^{k-1} + r_{i(k-1)}^{k-1}(r_{(k-1)(k-1)}^{k-1})^* r_{(k-1)j}^{k-1}$.

Falso, aunque se trata de una pregunta del Tema 2, la expresión correcta es:

$$r_{ij}^{k} = r_{ij}^{k-1} + r_{ik}^{k-1}(r_{kk}^{k-1})^{*}r_{kj}^{k-1}$$

17. Cuando se construye la expresión regular asociada a un autómata finito determinista, r_{ii}^{0} no puede ser nunca vacío.

Falso, sí puede serlo.

18. El conjunto de las palabras $\{u0011v^{-1} \mid u, v \in \{0,1\}^{*}\}$ es regular.

Verdadero, para verlo más claro (usando que puede $u \neq v$):

$$\{u0011v^{-1} \mid u, v \in \{0,1\}^{*}\} = \{u0011w \mid u, w \in \{0,1\}^{*}\}$$

Y podemos reconocer este lenguaje mediante la gramática lineal por la derecha $G = (\{S, A\}, \{0,1\}, P, S)$ con P el conjunto que contiene las siguientes producciones:

$$S \to 0S \mid 1S \mid 0011A$$
$$A \to 0A \mid 1A \mid \varepsilon$$

19. Si L es un lenguaje finito, entonces su complementario es siempre regular.

Verdadero, ya que si L es finito, entonces es regular, con lo que podemos construir un autómata finito determinista que reconozca dicho lenguaje. Una vez que obtengamos dicho autómata finito **determinista** $M = (Q, A, \delta, q_0, F)$, bastará considerar el autómata $M' = (Q, A, \delta, q_0, Q \setminus F)$, autómata finito determinista que aceptará el lenguaje $L(M') = \overline{L}$.

20. En un autómata finito determinista la relación de indistinguibilidad es una relación de equivalencia.

Verdadero, cumple las propiedades reflexiva, simétrica y transitiva, tal y como se ha visto en teoría.

21. En un autómata finito determinista siempre debe de existir, al menos, un estado de error.

Falso, el autómata que hicimos para la pregunta 14 que podemos ver en la Figura 1.105 era un autómata finito determinista totalmente válido y no tenía estados de error.

22. El conjunto de los números en binario que son múltiplos de 7 es regular.

Verdadero, como vimos en el Ejercicio 1.1.17.2 y en el AFD de la Figura 1.1.

23. Hay situaciones en las que los estados inaccesibles de un AFD cumplen una función específica.

    Falso, no se puede llegar nunca a un estado inaccesible, por lo que son irrelevantes en el reconocimiento de una palabra.

24. Si R es un lenguaje regular y L un lenguaje independiente del contexto, entonces R/L es regular.

    Verdadero, ya que no hace falta exigir hipótesis sobre L, sea cual sea dicho lenguaje, mientras que R sea regular, R/L será regular.

25. Si en un autómata dos estados son distinguibles de nivel n, entonces serán distinguibles de nivel m para todo $m \geqslant n$.

    Verdadero, ya que dos estados son distinguibles de nivel n si y solo si existe una palabra $u \in A^*$ de longitud menor o igual que n tal que en el conjunto $\{\delta^*(p,u), \delta^*(q,u)\}$ hay un estado final y otro no final, por lo que si p y q son distinguibles de nivel n por la dicha existencia de una palabra $u \in A^*$, entonces serán indistinguibles de nivel m para $m \geqslant n$, ya que podemos considerar la misma palabra que considerábamos para el caso de n, por ser $|u| \leqslant n \leqslant m$.

26. Si h es un homomorfismo y $h(L)$ no es regular, podemos concluir que L no es regular.

    Verdadero, es la implicación contrarrecíproca de "si h es un homomorfismo y L es regular, entonces $h(L)$ es regular", vista en teoría.

27. El lenguaje de todas las palabras en las que los tres primeros símbolos son iguales a los tres últimos es regular.

    Verdadero, supongamos que trabajamos sobre el alfabeto $A = \{a_1, \ldots, a_n\}$. Sabemos que hay un número finito de combinaciones para coger tres símbolos de dicho lenguaje: n^3 combinaciones distintas. Podemos pues, construir una aplicación biyectiva $f : \{1, 2, \ldots, n^3\} \to A \times A \times A$. De esta forma, podemos considerar la gramática:

    $$G = (\{S\} \cup \{A_i\}_{i \in \{1,2,\ldots,n^3\}}, A, P, S)$$

    Siendo P el conjunto de producciones que contienen las siguientes producciones que vamos a describir.

    S va a poder generar cada una de las n^3 sucesiones de 3 símbolos sobre A, cada una seguida de una variable A_i siendo i el índice de dicha combinación. Posteriormente, todas las variables A_i podrán generar un número indefinido de ceros y unos en cualquier orden, teniendo que terminar con la combinación

de los 3 símbolos correspondiente al índice i:

$$\begin{aligned} S &\to f(1)A_1 \mid f(2)A_2 \mid \ldots \mid f(n^3)A_{n^3} \\ A_1 &\to 0A_1 \mid 1A_1 \mid f(1) \\ A_2 &\to 0A_2 \mid 1A_2 \mid f(2) \\ &\vdots \\ A_i &\to 0A_i \mid 1A_i \mid f(i) \\ &\vdots \\ A_{n^3} &\to 0A_{n^3} \mid 1A_{n^3} \mid f(n^3) \end{aligned}$$

Y tenemos que G es una gramática regular por la derecha, por lo que el lenguaje que genera es regular.

28. Si un lenguaje verifica la condición que aparece en el lema de bombeo para lenguajes regulares, ya no hay forma de demostrar que no es regular.

    Falso, el lenguaje puede verificar la condición del lema de bombeo y no ser regular.

29. Si f es un homomorfismo entre alfabetos $f : A_1^* \to A_2^*$ y $L \subseteq A_1^*$ no es regular, podemos concluir que $f(L)$ tampoco es regular.

30. Todo lenguaje que cumple la condición del lema de bombeo para lenguajes regulares puede ser aceptado por un autómata finito no determinista.

    Falso, anteriormente comentamos que un lenguaje puede verificar la condición del lema de bombeo y no ser regular.

31. No existe algoritmo para saber si el lenguaje generado por una gramática regular es finito.

    Falso, sí existe. En primer lugar, hemos de eliminar los estados inaccesibles y los estados desde los que no se puede llegar a un estado final (estados de error). Una vez hecho esto, si el autómata finito determinista resultante tiene ciclos, entonces el lenguaje es infinito, ya que podemos ir dando vueltas por el ciclo y generando palabras infinitas. Si no hay ciclos, entonces el lenguaje es finito.

32. Dos autómatas finitos deterministas con diferente número de estados y que aceptan el lenguaje vacío tienen el mismo número de estados finales.

    Falso, solo sería cierto si todos los estados son alcanzables, ya que los autómatas de las Figuras 1.107 y 1.108 ambos aceptan el lenguaje vacío y el número de estados finales es distinto.

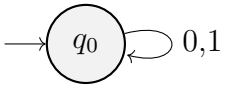


Figura 1.107: Autómata finito determinista 1 para la pregunta 32.

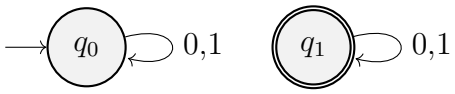


Figura 1.108: Autómata finito determinista 2 para la pregunta 32.

33. Si A es un alfabeto y L un lenguaje cualquiera distinto del vacío, entonces se verifica que $A^*/L = A^*$.

    Verdadero, si recordamos la definición de A^*/L:

    $$A^*/L = \{u \in A^* \mid \exists v \in L \text{ verificando } uv \in A^*\}$$

    $\subseteq$) Es trivial, por ser A^*/L un lenguaje.

    $\supseteq$) Sea $u \in A^*$, como $L \neq \emptyset$, existirá $v \in L$, con lo que $uv \in A^*$ por ser el conjunto de todas las palabras, con lo que $u \in A^*/L$.

34. Si R_{ij}^k son los lenguajes que se usan en la construcción de una expresión regular a partir de un autómata finito, siempre se verifica que $R_{ij}^{i-1} R_{jk}^{j-1} \subseteq R_{ik}^j$.

35. El lema de bombeo es útil para demostrar que la intersección de dos lenguajes regulares no es regular.

36. Existe un algoritmo para determinar si el lenguaje generado por una gramática regular es infinito.

37. Existe un algoritmo para determinar si el lenguaje generado por una gramática regular es finito o infinito.

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

    Verdadero, ya que la intersección de dos lenguajes regulares da lugar a un lenguaje regular (tal y como veíamos en la pregunta 3), que a su vez es independiente del contexto.

39. Si un lenguaje es infinito no se puede encontrar una expresión regular que lo represente.

Falso, debido a la existencia de lenguajes regulares infinitos. Por ejemplo, podemos considerar $L = \{a^i \mid i \in \mathbb{N}\}$, un lenguaje infinito por ser biyectivo con $\mathbb{N}$:

$$L = \{\varepsilon, a, aa, aaa, aaaa, \ldots\}$$

Y este puede ser representado por la expresión regular $(a)^*$.

40. En un autómata finito determinista sin estados inaccesibles la relación de indistiguibilidad entre los estados es una relación de equivalencia.

Verdadero, tal y como vimos en teoría.

41. En un autómata finito determinista, si no hay dos estados que sean indistinguibles entre sí, entonces el autómata es minimal.

Falso, salvo si el autómata no tiene estados inalcanzables, en cuyo caso es verdadero.

42. Dada una gramática lineal por la derecha, siempre existe otra gramática lineal por la izquierda que acepte el mismo lenguaje.

Verdadero, tal y como vimos en la teoría del Tema 2.

43. Si R es un lenguaje regular y L un lenguaje cualquiera, entonces R/L es siempre un lenguaje regular.

Verdadero, tal y como hemos visto en la teoría.

44. Si un lenguaje cumple la condición del lema de bombeo para conjuntos regulares no nos asegura que sea un lenguaje regular.

Verdadero, ya que la propiedad del lema de bombeo es una condición necesaria, no suficiente.

45. Existe un algoritmo para determinar si los lenguajes generados por dos gramáticas regulares son iguales o no.

Verdadero. En teoría hemos visto que si tenemos dos lenguajes, L_1 y L_2 generados por dos autómatas finitos de terministas, M_1 y M_2 (respectivamente), entonces podemos construir el autómata finito para el lenguaje diferencia simétrica $L_1 \ \Delta \ L_2 = (L_1 \setminus L_2) \cup (L_2 \setminus L_1)$, y aplicarle el algoritmo de si el lenguaje que genera es vacío (en cuyo caso, $L_1 = L_2$). Además, como tenemos un algoritmo para pasar de gramáticas regulares a autómatas, podemos mecanizar todo este proceso.

46. El conjunto de cadenas aceptado por un autómata finito no determinista con transiciones nulas no puede ser generado por una gramática independiente del

contexto.

Falso, el conjunto de cadenas acpetadas por un autómata finito no determinista con transiciones nulas es un lenguaje regular, que puede ser generado por una gramática regular por la derecha, que a su vez es independiente del contexto.

47. El lenguaje resultado de la unión de dos lenguajes regulares con un número infinito de palabras puede ser representado mediante una expresión regular.

Verdadero, ya que la unión de dos lenguajes regulares es regular, con lo que puede ser representado mediante una expresión regular, independientemente de la cardinalidad de los lenguajes.

48. Una expresión regular siempre representa a un lenguaje que puede ser generado por una gramática independiente del contexto.

Verdadero, ya que una expresión regular siempre representa a un lenguaje que puede ser generado por una gramática regular por la derecha, que a su vez es independiente del contexto.

49. Existe un algoritmo para comprobar si son iguales los lenguajes aceptados por dos autómatas finitos diferentes.

Verdadero, en caso de ser autómatas finitos determinisitas, es el razonamiento que ya hicimos en la pregunta 45. En caso de ser autómatas finitos no deterministas, existe un algoritmo para pasarlos a deterministas y en cuyo caso, podemos aplicar el algoritmo ya mencionado.

50. Si en un autómata finito no determinista intercambio entre sí los estados finales y no finales obtengo un autómata que acepta el lenguaje complementario del aceptado por el autómata original.

Falso, esta pregunta ya fue respondida anteriormente, en la pregunta 9.

51. Si L es un lenguaje regular, entonces el lenguaje LL^{-1} es también regular.

Verdadero, ya que si L es regular, entonces L^{-1} es regular; y la concatenación de lenguajes regulares sigue siendo regular.

52. El lema de bombeo para lenguajes regulares es útil para demostrar que un lenguaje determinado no es regular.

Verdadero, puede ser útil para demostrar que un lenguaje no es regular, aunque puede haber casos en los que no nos dé información sobre si es regular o no, al tratarse de una condición necesaria para los lenguajes regulares.

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

    Falso, el lenguaje $L = \{a^i \mid i \in \mathbb{N}\}$ tiene un conjunto infinito de palabras (es biyectivo con $\mathbb{N}$) y es regular, ya que puede representarse mediante la expresión regular $(a)^*$.

54. Un autómata finito determinista sin estados inaccesibles ni indistinguibles es minimal.

    Verdadero, tal y como comentamos anteriormente en la pregunta 12.

55. El conjunto de las palabras $\{u0011v^{-1} \mid u, v \in \{0,1\}^*\}$ es regular.

    Verdadero, tal y como comentamos anteriormente en la pregunta 18.

56. Existe un algoritmo para determinar si el lenguaje generado por una gramática regular es infinito.

    Verdadero, como razonamos en la pregunta 31.

57. Para cada autómata finito no determinista M existe una gramática independiente de contexto G tal que $\mathcal{L}(M) = \mathcal{L}(G)$.

    Verdadero, ya que sabemos que para cada autómata finito no determinista M existe una gramática regular por la derecha G tal que $\mathcal{L}(M) = \mathcal{L}(G)$ y sabemos que las gramáticas regulares por la derecha son a su vez independientes del contexto.

58. 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 construir un autómata finito determinista que acepte el lenguaje, tal y como vemos en la Figura 1.109.

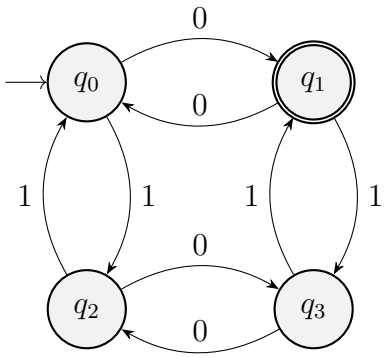


Figura 1.109: Autómata finito determinista para la pregunta 58.

Donde pensamos en los estados como:

- q_0, la palabra tiene un número par de ceros y de unos.
- q_1, la palabra tiene un número impar de ceros y número par de unos.
- q_2, la palabra tiene un número par de ceros y número impar de unos.
- q_3, la palabra tiene un número impar de ceros y de unos.

Elegimos como estado inicial q_0 ya que la palabra ε tiene un número par de ceros y de unos.

59. Si L es un lenguaje regular, entonces la cabecera de L (CAB(L)) es siempre regular.

Verdadero, si L es un lenguaje regular, existirá un autómata finito determinista $M = (Q, A, \delta, q_0, F)$ que reconozca dicho lenguaje. Consideramos ahora el autómata $M' = (Q, A, \delta, q_0, F')$, un autómata igual que M con la diferencia de que $q \in F'$ si y solo si existen dos palabras $u, v \in A^*$ tales que:

$$\delta^*(q_0, u) = q \quad \wedge \quad \delta^*(q, v) \in F$$

Es decir, si q es un estado alcanzable y se encuentra en el camino previo a un estado final, de forma que la palabra que se lee desde q_0 hasta q será un prefijo de alguna palabra de L.

60. En un autómata finito determinista, si no hay dos estados que sean indistinguibles entre si, entonces el autómata es minimal.

Falso, es necesario exigir además que no haya estados inalcanzables.

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

Verdadero, ya que la intersección de dos lenguajes regulares da lugar a un lenguaje regular, que a su vez es independiente del contexto.

62. Si un lenguaje es infinito no se puede encontrar una expresión regular que lo represente.

Falso, pregunta que ya fue contestada en la pregunta 39.

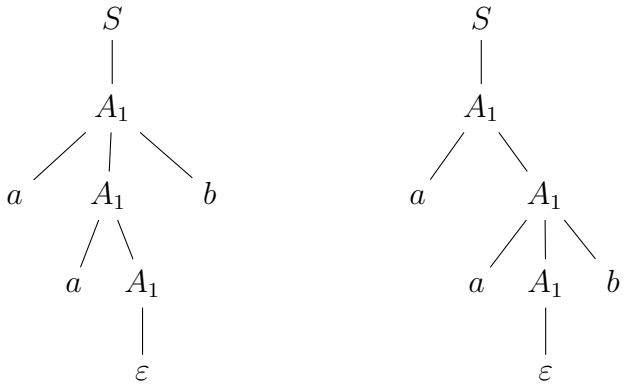


Figura 1.110: Árboles de derivación para *aab* usando la Gramática del Ejercicio 1.4.1.

1.4. Gramáticas Independientes del Contexto

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

Ejercicio 1.4.1. Determinar si la siguiente gramática es ambigua y si el lenguaje generado es inherentemente ambiguo:

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

La gramática dada es ambigua puesto que hay palabras con más de un árbol de derivación. Por ejemplo, la palabra *aab* tiene los dos posibles árboles de derivación que se muestran en la Figura 1.110.

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

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

Este lenguaje es regular con expresión regular asociada:

$$a^*b^*$$

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

Ejercicio 1.4.2. Sea la gramática

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

1. Demostrar que es ambigua.

   Tomemos como palabra *aabb*. Esta palabra tiene dos árboles de derivación, como se muestra en la Figura 1.111.

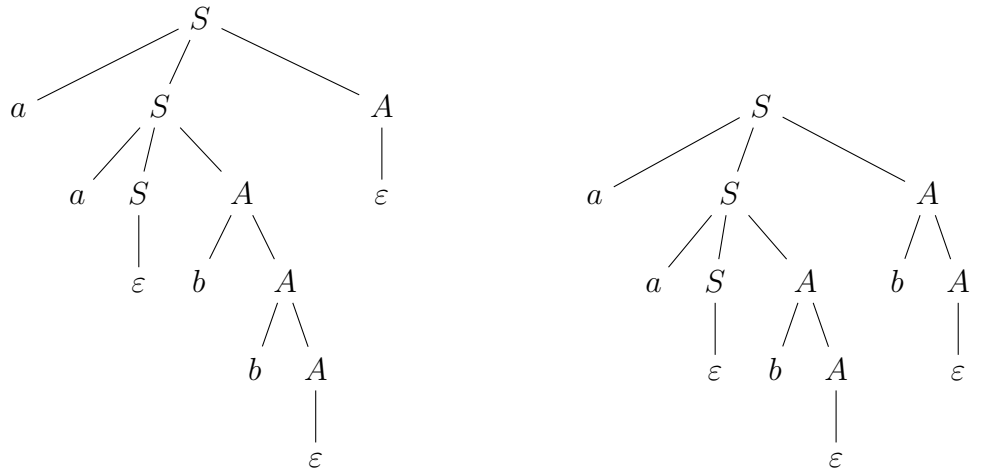


Figura 1.111: Árboles de derivación para *aabb* usando la Gramática del Ejercicio 1.4.2.

2. Dar una expresión regular para el lenguaje generado.

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

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

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

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

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

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

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

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

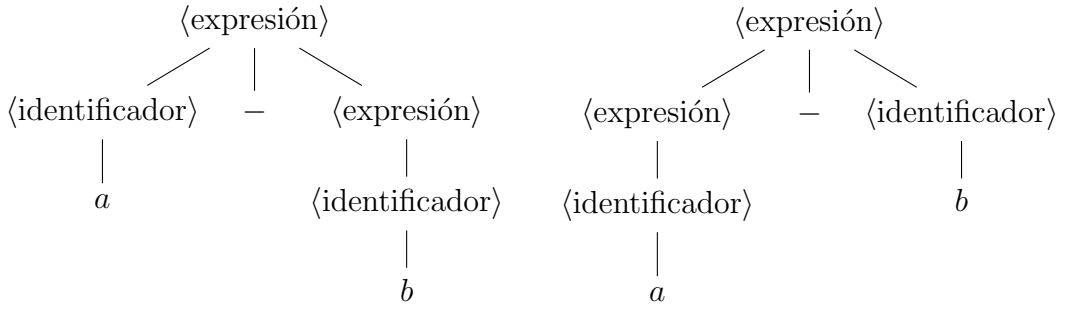


Figura 1.112: Árboles de derivación para $a - b$ usando la Gramática del Ejercicio 1.4.3.

Ejercicio 1.4.3. Considera la gramática $G = (V, T, S, P)$ donde

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

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

   El lenguaje generado tiene como expresión regular:

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

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

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

   Para ver que la gramática es ambigua, consideramos la palabra $a - b$. Esta palabra tiene dos árboles de derivación, como se muestra en la Figura 1.112.

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

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

   Sí, podemos añadir paréntesis a la gramática para que sea capaz de generar el lenguaje de las expresiones aritméticas correctas con el operador de resta. La

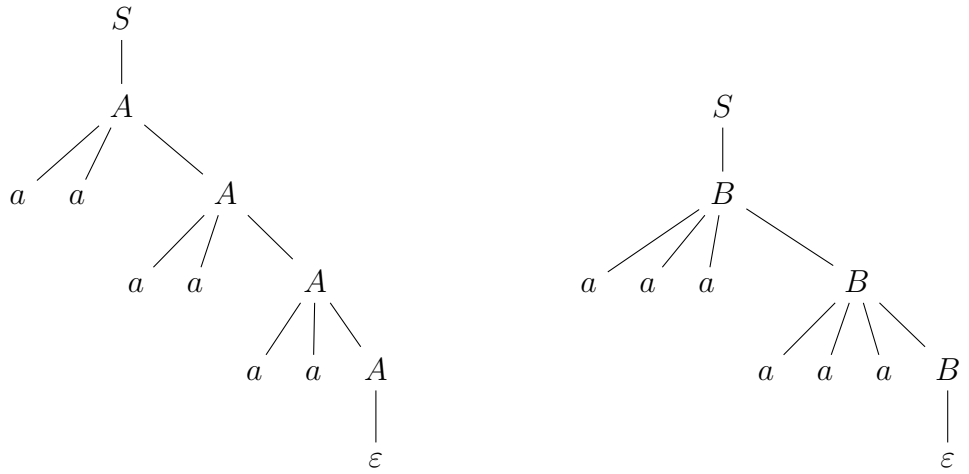


Figura 1.113: Árboles de derivación para a^6 usando la Gramática del Ejercicio 1.4.4.

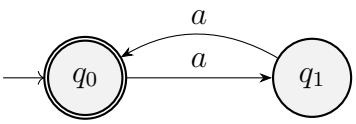


Figura 1.114: AFD que acepta el lenguaje de la variable A de la Gramática del Ejercicio 1.4.4.

gramática modificada sería $G = (V, \{a, b, c, d, -, (,)\}, P, S)$, con:

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

Ejercicio 1.4.4. Dada la gramática

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

1. Demostrar que es ambigua.

   La variable A genera palabras de la forma a^{2i} y la variable B genera palabras de la forma a^{3i}. Por tanto, la palabras de la forma a^{6i} tienen dos árboles de derivación, como se muestra en la Figura 1.113.

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

   El autómata que genera las palabras de la forma a^{2i} es el que se muestra en la Figura 1.114, mientras que el autómata que genera las palabras de la forma a^{3i} es el que se muestra en la Figura 1.115. El autómata producto es el que se muestra en la Figura 1.116.

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

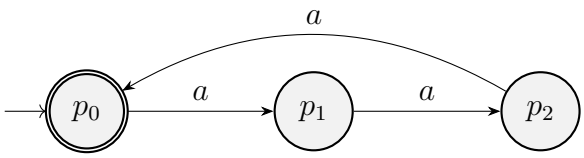


Figura 1.115: AFD que acepta el lenguaje de la variable B de la Gramática del Ejercicio 1.4.4.

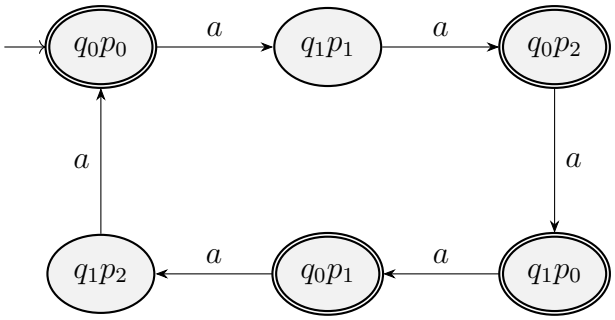


Figura 1.116: AFD que acepta el lenguaje de la Gramática del Ejercicio 1.4.4.

La gramática lineal por la derecha que genera el lenguaje del autómata de la Figura 1.116 es $G = (V, \{a\}, P, S)$, con:

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

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

4. Demostrar que la gramática resultante no es ambigua.

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

Ejercicio 1.4.5. Dar una gramática libre de contexto no ambigua que genere el lenguaje

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

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

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

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

Por tanto, tenemos que:

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

Damos ahora una gramática para cada uno de los lenguajes.

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

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

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

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

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

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

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

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

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

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

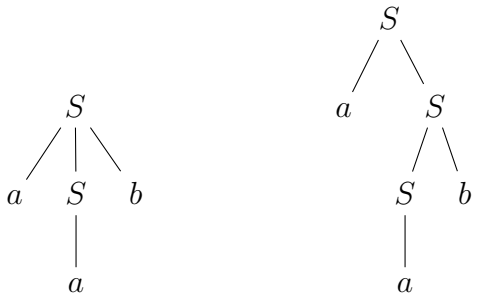


Figura 1.117: Árboles de derivación para aab usando la Gramática del Ejercicio 1.4.6.1.

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

$$V_3 = \{S_3, C_3\}$$
$$P_3 = \begin{cases} S_3 \to C_3C_3 \mid \varepsilon \\ C_3 \to aC_3b \mid ab \end{cases}$$

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

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

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

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

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

Ejercicio 1.4.6. Determinar cuales de las siguientes gramáticas son ambiguas y, en su caso, comprobar si los lenguajes generados son inherentemente ambiguos:

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

   La gramática dada es ambigua. Por ejemplo, la palabra aab tiene dos árboles de derivación, como se muestra en la Figura 1.117. No obstante, este es un lenguaje regular con expresión regular asociada:

   $$aa^*b^*$$

   Por tanto, el lenguaje generado no es inherentemente ambiguo.

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

   La gramática dada es ambigua. Por ejemplo, la palabra a^7 tiene dos árboles de derivación, como se muestra en la Figura 1.118. No obstante, este es un lenguaje regular con expresión regular asociada:

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

   Por tanto, el lenguaje generado no es inherentemente ambiguo.

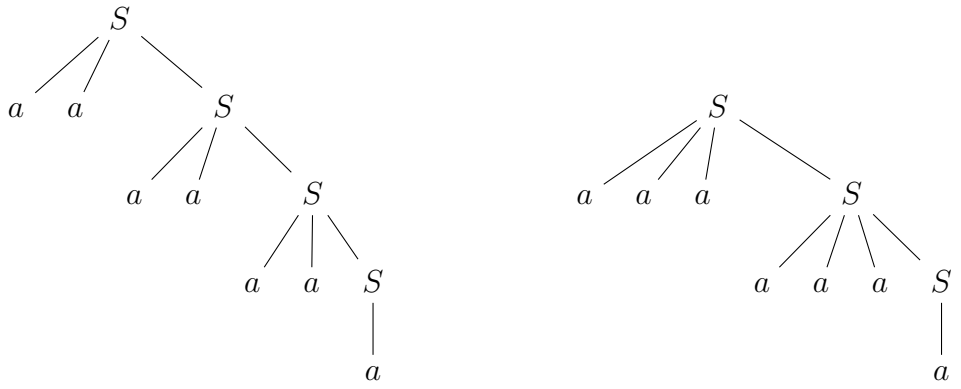


Figura 1.118: Árboles de derivación para a^7 usando la Gramática del Ejercicio 1.4.6.2.

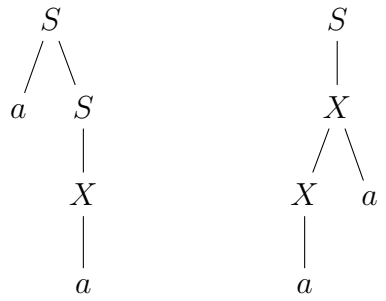


Figura 1.119: Árboles de derivación para a^2 usando la Gramática del Ejercicio 1.4.6.3.

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

   La gramática dada es ambigua. Por ejemplo, la palabra a^2 tiene dos árboles de derivación, como se muestra en la Figura 1.119.

   El lenguaje generado por esta gramática es:

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

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

$$V = \{S, A, B\}$$

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

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

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

Ejercicio 1.4.7. Dar gramáticas libres de contexto no ambiguas (cuando sea posible) pa2ra los siguientes lenguajes sobre el alfabeto $A = \{a, b, c\}$:

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

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

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

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

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

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

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

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

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

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

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

   Notemos que, como el enunciado no es totalmente preciso, consideramos $aabbc \notin L_4$. El AFD que acepta el lenguaje L_4 es el de la Figura 1.120.

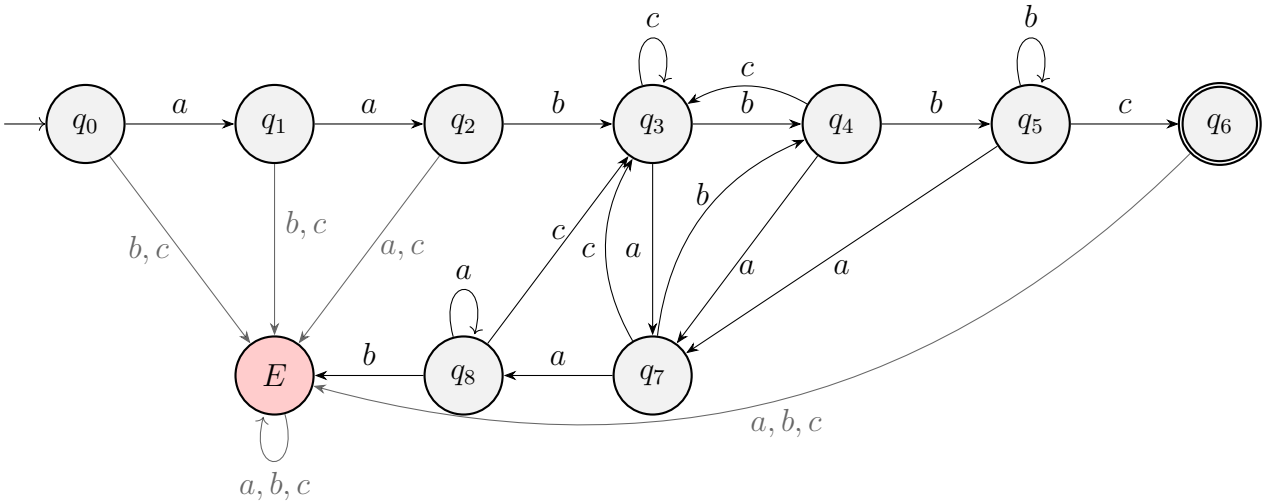


Figura 1.120: AFD que acepta el lenguaje L_4.

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

$$V = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$$

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

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

Ejercicio 1.4.8. Dada la gramática

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

1. Determinar si es ambigua.

   La gramática dada es ambigua. Por ejemplo, la palabra 010101 tiene dos árboles de derivación, como se muestra en la Figura 1.121.

2. Construir un autómata finito determinista asociado.

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

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

   El AFND asociado es el de la Figura 1.122.

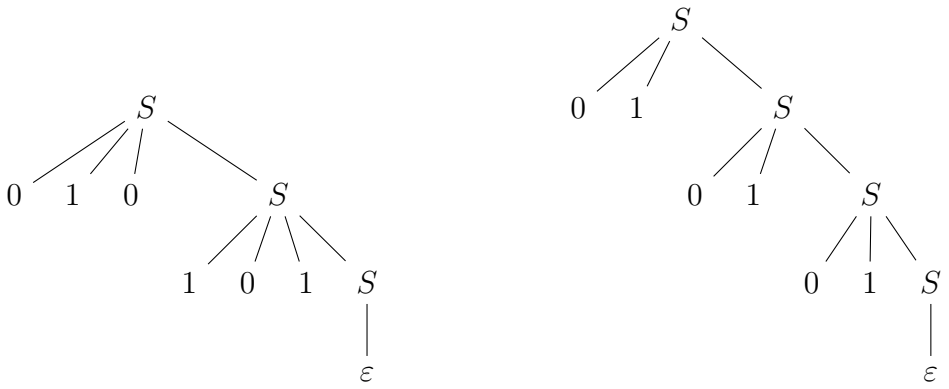


Figura 1.121: Árboles de derivación para 010101 usando la Gramática del Ejercicio 1.4.8.

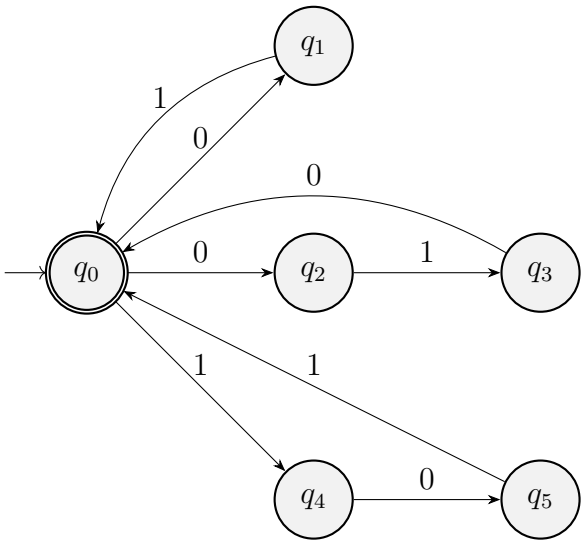


Figura 1.122: AFND que acepta el lenguaje de la Gramática del Ejercicio 1.4.8.

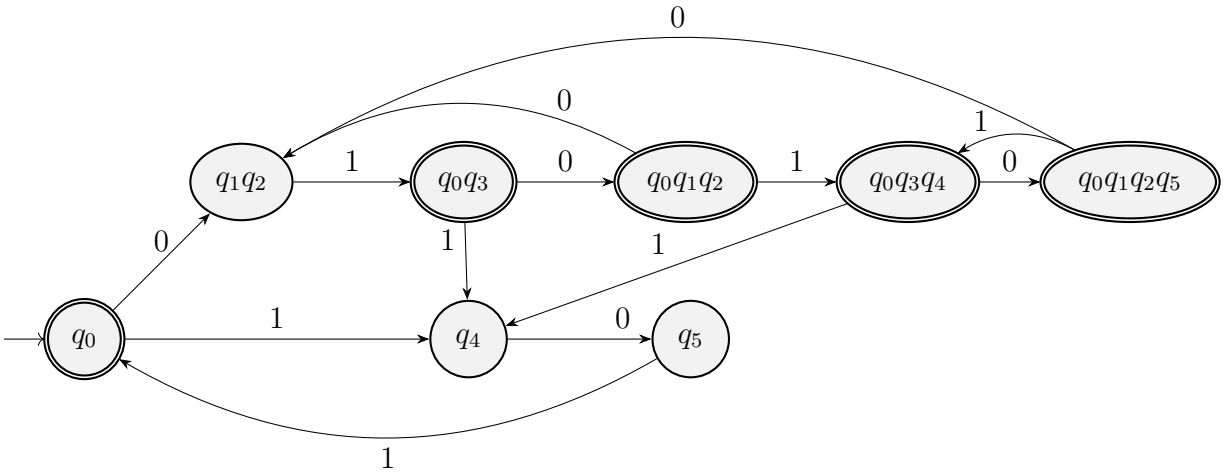


Figura 1.123: AFD que acepta el lenguaje de la Gramática del Ejercicio 1.4.8.

El AFD asociado es el de la Figura 1.123, donde las transiciones que faltan son transiciones a un estado de error. Este no se añade para evitar confusión, y no afectará al siguiente apartado.

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

   La gramática lineal por la derecha que se obtiene a partir del AFD es $G = (V, \{0,1\}, P, q_0)$, con:

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

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

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

Ejercicio 1.4.9. Considerar la siguiente gramática:

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

1. Demostrar que la gramática dada no es ambigua.

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

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

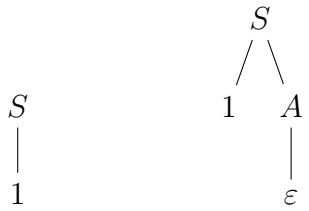


Figura 1.124: Árboles de derivación para 1 usando la Gramática del Ejercicio 1.4.9.

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

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

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

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

   Notemos que esta es ambigua, puesto que si consideramos la palabra 1, esta tiene dos árboles de derivación, como se muestra en la Figura 1.124. Notemos que si quitamos la regla $S \to 1$, la gramática no sería ambigua y generaría el mismo lenguaje que la gramática original.

Ejercicio 1.4.10. Considerar la siguiente gramática $G = (\{S, A\}, \{a, b\}, P, S)$, con

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

1. Describir el lenguaje generado por la gramática.

   Tenemos que:

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

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

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

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

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

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

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

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

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

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

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

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

Ejercicio 1.4.11. Obtener la forma normal de Greibach para la siguiente gramática:

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

donde:

$$P = \begin{cases} S_1 \to S_1S_2c \mid S_3 \mid S_3bS_3 \\ S_2 \to S_1S_1 \mid d \\ S_3 \to S_2e \end{cases}$$

Vemos que esta gramática no tiene producciones nulas. Eliminamos las producciones unitarias. Para esto, vemos que las únicas producciones unitarias son $S_1 \to S_3$ y $S_3 \to S_2e$. Por tanto, eliminamos estas producciones unitarias y obtenemos las siguientes reglas de producción:

$$P = \begin{cases} S_1 \to S_1S_2c \mid S_2e \mid S_3bS_3 \\ S_2 \to S_1S_1 \mid d \\ S_3 \to S_2e \end{cases}$$

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

resultantes son:

$$P = \begin{cases} S_1 \to S_1S_2C_c \mid S_2C_e \mid S_3C_bS_3 \\ S_2 \to d \mid S_1S_1 \\ S_3 \to S_2C_e \\ C_b \to b \\ C_c \to c \\ C_e \to e \end{cases}$$

Eliminamos ahora las producciones de la forma $A \to B_1B_2 \dots B_n$, con $n \geqslant 3$. Para esto, introducimos variables auxiliares. Las producciones resultantes son:

$$P = \begin{cases} S_1 \to S_1D_1 \mid S_2C_e \mid S_3D_2 \\ S_2 \to d \mid S_1S_1 \\ S_3 \to S_2C_e \\ C_b \to b \\ C_c \to c \\ C_e \to e \\ D_1 \to S_2C_c \\ D_2 \to C_bS_3 \end{cases}$$

Llegados a este punto, tenemos la gramática en Forma Normal de Chomsky, por lo cual ya podemos aplicar el algoritmo para la obtención de la Forma Normal de Greibach.

Ejercicio 1.4.12. Pasar a forma normal de Greibach la gramática

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

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

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

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

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

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

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

Ejercicio 1.4.13. Determina si los siguientes lenguajes son regulares o independientes del contexto. Encuentra una gramática que los genere.

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

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

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

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

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

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

   a) L_{aux}. Se da la igualdad. Cada c está balanceada con un a o un b, y cada a y cada b están balanceados con un c.
   b) L_{1a}. $k < i + j$. Cada c está balanceada con un a o un b. Cada b está balanceada con un c, y al menos un a no está balanceado con un c.
   c) L_{1b}. $k < i + j$. Cada c está balanceada con un a o un b. Cada a está balanceada con un c, y al menos un b no está balanceado con un c.
   d) L_{1ab}. $k < i + j$. Cada c está balanceada con un a o un b. Al menos un a no está balanceado con un c, y al menos un b no está balanceado con un c.

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

   Lenguaje auxiliar L_{aux}) Este lenguaje es:

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

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

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

   Esta gramática no nos es estrictamente necesaria, pero nos será de ayuda para obtener las gramáticas que generan los lenguajes L_{1a}, L_{1b} y L_{1ab}.

Notemos que $S \to aSc$ genera un a balanceada con un c, y $X \to bXc$ genera un b balanceado con un c. Estas dos reglas habrán de mantenerse para que cada c esté balanceada con un a o un b.

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

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

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

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

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

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

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

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

A modo de resumen, y reduciendo el número de variables, la gramática que

genera L_1 es:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

puesto que no comienza por d.

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

La gramática que genera L_2 es:

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

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

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

   Dado $z \in L_3$, tenemos que:

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

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

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

   Por tanto, la gramática que genera L_3 es:

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

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

Consideramos la gramática que genera L_2:

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

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

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

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

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

Ejercicio 1.4.14. Dadas las siguientes gramáticas determinar si son ambiguas y, en caso de que lo sean, determinar una gramática no ambigua que genere el mismo lenguaje.

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

   Esta gramática es ambigua. Por ejemplo, la palabra $x+y+x$ tiene dos árboles de derivación, como se muestra en la Figura 1.125.

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

Ejercicio 1.4.15. Una gramática independiente del contexto generalizada es una gramática en el que las producciones son de la forma $A \to r$ donde r es una expresión regular de variables y símbolos terminales. Una gramática independiente del contexto generalizada representa una forma compacta de representar una gramática con todas las producciones $A \to \alpha$, donde α es una palabra del lenguaje asociado a la expresión regular r y $A \to r$ es una producción de la gramática generalizada. Observemos que esta gramática asociada puede tener infinitas producciones, ya que una expresión regular puede representar un lenguaje con infinitas palabras. El concepto

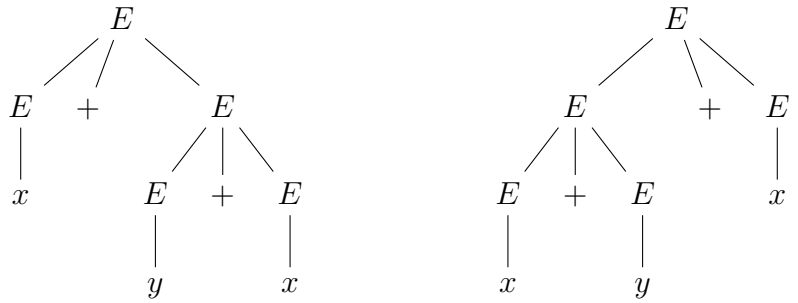


Figura 1.125: Árboles de derivación para $x + y + x$ usando la Gramática del Ejercicio 1.4.14.

de lenguaje generado por una gramática generalizada se define de forma análoga al de las gramáticas independientes del contexto, pero teniendo en cuenta que ahora puede haber infinitas producciones. Demostrar que un lenguaje es independiente del contexto si y solo si se puede generar por una gramática generalizada.

Demostraremos por doble implicación.

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

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

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

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

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

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

- Si $r = \emptyset$, eliminamos esa regla de producción.
- Si $r = \varepsilon$, esta es una regla aceptada en una gramática independiente del contexto.
- Si $r = a$, con $a \in T$, esta es una regla aceptada en una gramática independiente del contexto.
- Si $r = s + t$, con s, t expresiones regulares, entonces la eliminamos y añadimos la regla $A \to s \mid t$.
- Si $r = st$, con s, t expresiones regulares, entonces la eliminamos y añadimos las variables auxiliares $\{A_1, A_2\}$ y las reglas $A \to A_1A_2$, $A_1 \to s$, $A_2 \to t$.

- Si $r = s^*$, con s expresión regular, entonces la eliminamos y añadimos la variable auxiliar $\{B\}$ y las reglas $A \to B$, $B \to sB \mid \varepsilon$, donde la B ha sido necesaria para asegurarse de que se cumple la clausura de Kleene.

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

Ejercicio 1.4.16. Demostrar que los siguientes lenguajes son independientes del contexto:

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

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

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

   Notemos que:

   - La variable S genera Av_2.
   - La variable A genera uBu^{-1}.
   - La variable B genera $\#v_1$.

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

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

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

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

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

   De esta forma, tenemos:

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

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

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

De esta forma, tenemos:

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

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

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

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

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

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

Ejercicio 1.4.17. Sobre el alfabeto $\{0, 1\}$ dar una gramática no ambigua que genere todas las palabras en las que el número de 0s es el doble que el de 1s.

Ejercicio 1.4.18. Sea el lenguaje $L = \{u\#v \mid u, v \in \{0, 1\}^*,\ u \neq v\}$. Demostrar que L es independiente del contexto.

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

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

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

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

Distinguimos en función de u, v:

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

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

  Buscamos genarar entonces palabras de la forma $u\#v$ con $n \neq m$. En este caso, alguna de las dos palabras tiene más símbolos que la otra, por lo que no

hemos de preocuparnos de que sean distintas, ya que esto se tiene al ser de distinta longitud. Estas palabras se generan con:

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

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

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

$$\underbrace{a_1 \cdots a_{i-1}}_{3^\circ}\underbrace{a_i}_{4^\circ}\underbrace{a_{i+1} \cdots a_n}_{5^\circ}\underbrace{\#}_{6^\circ}\underbrace{b_1 \cdots b_{i-1}}_{3^\circ}\underbrace{b_i}_{2^\circ}\underbrace{b_{i+1} \cdots b_m}_{1^\circ}$$

$$a_p, b_q \in \{0,1\}\ \forall p \in \{1, \ldots, n\},\ \forall q \in \{1, \ldots, m\},\ m, n \in \mathbb{N}, a_i \neq b_i$$

  Las reglas de producción son las siguientes:

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

Por tanto, la gramática que genera L es:

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

Ejercicio 1.4.19. Demostrar que si una gramática G está en forma normal de Chomsky, entonces si $w \in \mathcal{L}(G)$ el número de pasos de derivación de toda generación de esta palabra es $2|w| - 1$.

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

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

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

$$\begin{aligned} A &\to BC \qquad B, C \in V \\ A_i &\to a_i \qquad a_i \in T \end{aligned}$$

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

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

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

Ejercicio 1.4.20. Dar gramáticas independientes del contexto no ambiguas para los siguientes lenguajes sobre el alfabeto $\{0, 1\}$:

1. L_1, El conjunto de palabras w tal que en todo prefijo de w el número de 0s es mayor o igual que el número de 1s.

   Sea $w \in L_1$, con $|w| = n \in \mathbb{N}$. Para cada $i \in \{1, \ldots, n\}$ notamos:

   $$w_i = a_1 a_2 \cdots a_i, \qquad w = a_1 a_2 \cdots a_n$$

   Como $w \in L_1$, es necesario que $N_0(w_i) \geqslant N_1(w_i)\ \forall i \in \{1, \ldots, n\}$.

2. L_2, El conjunto de palabras w en las que el número de 0s es mayor o igual que el número de 1s.

Ejercicio 1.4.21. Sea $L = \{0^i 1^j 0^k \mid i, j, k \in \mathbb{N} \cup \{0\},\ i \neq j, 2i \neq j\}$. Demostrar que L es independiente del contexto.

Ejercicio 1.4.22. Supongamos la siguiente gramática:

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

1. Demostrar que la gramática es ambigua.

   La siguiente gramática genera el mismo lenguaje que la anterior:

$$
\begin{aligned}
&G = (V, T, P, \langle \text{SENT} \rangle) \\
&V = \{\langle \text{SENT} \rangle\} \\
&T = \{\text{if}, \text{condicion}, \text{then}, \text{else}, a := 1\} \\
&P = \begin{cases} \langle \text{SENT} \rangle \to \text{a} := 1 \\ \langle \text{SENT} \rangle \to \text{if condicion then } \langle \text{SENT} \rangle \\ \langle \text{SENT} \rangle \to \text{if condicion then } \langle \text{SENT} \rangle \text{ else } \langle \text{SENT} \rangle \end{cases}
\end{aligned}
$$

   Una palabra que pertenece al lenguaje con más de un árbol de derivación es:

$$\text{if condicion then if condicion then a} := 1 \text{ else a} := 1$$

   Ambos árboles de derivación se muestran en la Figura 1.126.

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

1.4.1. Preguntas Tipo Test

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

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

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

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

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

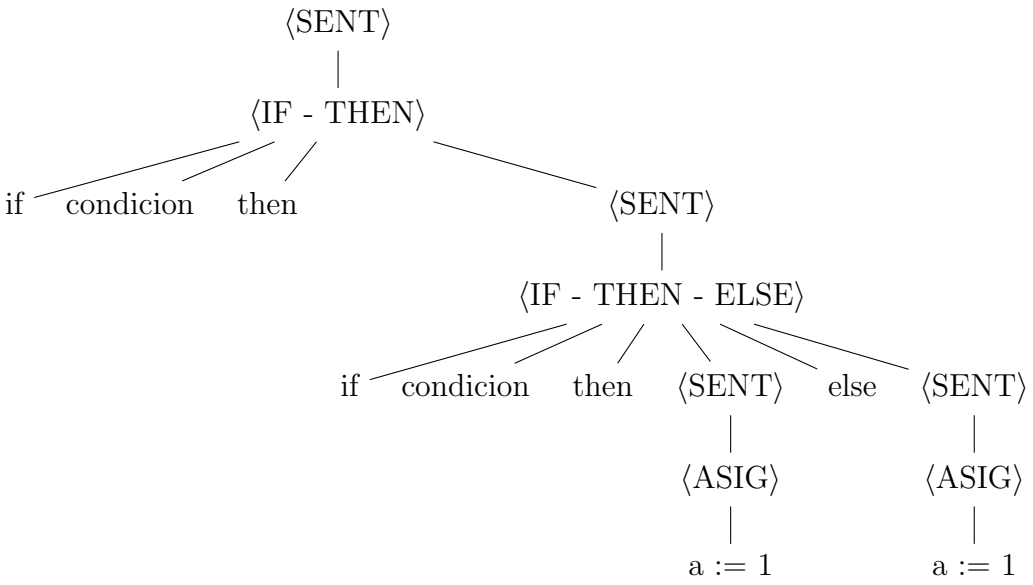
5. Un lenguaje inherentemente ambiguo puede ser generado por una gramática ambigua.

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

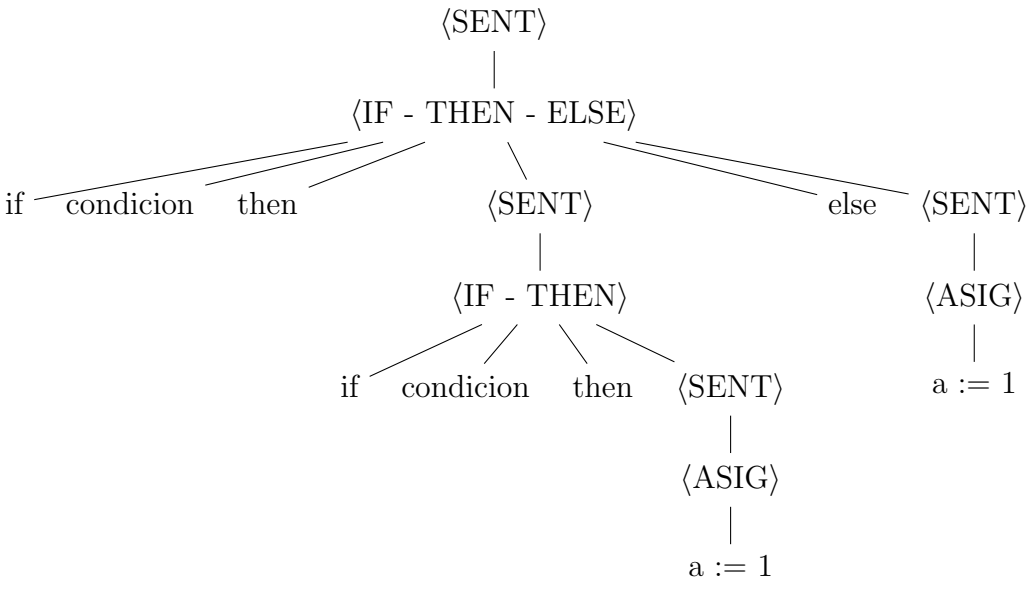
7. Si en una producción de una gramática independiente del contexto, uno de los símbolos que contiene es útil, entonces la producción es útil.

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

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



(a) Árbol de derivación 1.



(b) Árbol de derivación 2.

Figura 1.126: Árboles de derivación para la palabra `if condicion then if condicion then a := 1 else a := 1`.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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