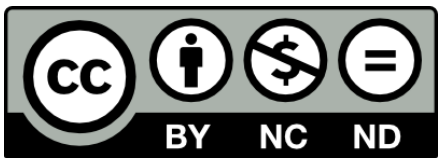
Modelos de Computación

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 $L = \mathcal{L}(G)$.

$\subset$) 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 $S \overset{*}{\Longrightarrow} ubbbv$.

$\supset$) 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 $L = \mathcal{L}(G)$.

$\subset$) 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, $w \in \mathcal{L}(G)$.

$\supset$) 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 $L = \mathcal{L}(G)$.

$\subset$) 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, $z \in \mathcal{L}(G)$.

$\supset$) 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 $L^* = \mathcal{L}(G)$.

$\subset$) 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 $X \to \varepsilon$.

$\supset$) 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 $L = \mathcal{L}(G)$.

$\subset$) 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 $w \in \mathcal{L}(G)$.

$\supset$) 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 $L = \mathcal{L}(G)$.

$\subset$) 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 $w \in \mathcal{L}(G)$.

$\supset$) 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 $L = \mathcal{L}(G)$.

$\subset$) 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 $w \in \mathcal{L}(G)$.

$\supset$) 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:

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

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

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

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

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:

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

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

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

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

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

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

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:

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

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 $L^* = L$.

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

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

Demostración. Demostramos mediante doble implicación.

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

Sea $v \in L^*$:

1. Si $v = \varepsilon \Longrightarrow v \in L$ 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 $u, v \in L = L^* \Longrightarrow uv \in L^* = L$.

□

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

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

□

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:

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

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:

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

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:

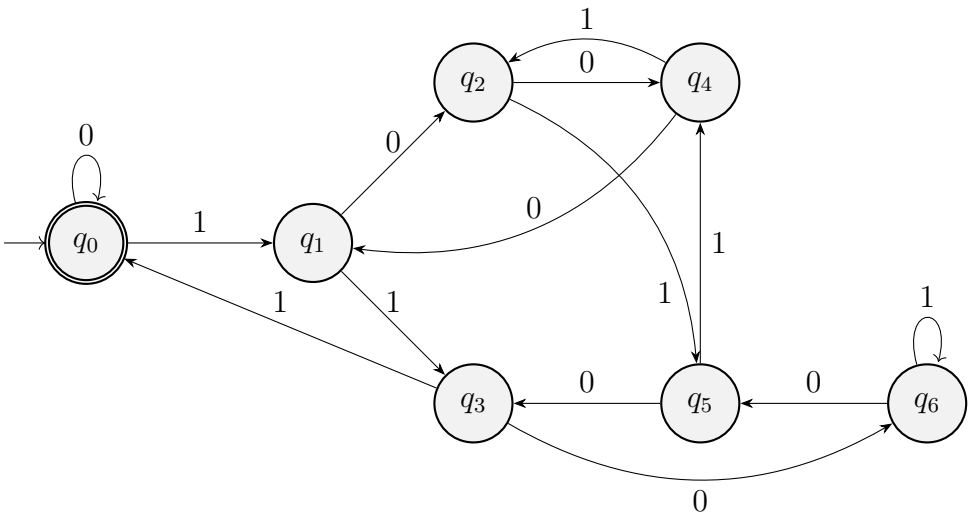


Figura 1.1: AFD que acepta los múltiplos de 7 en binario.

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.

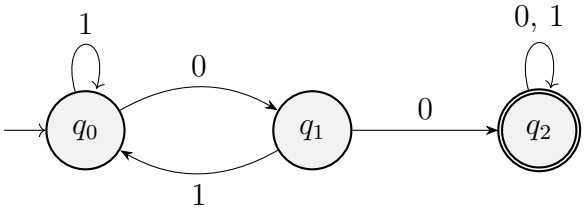


Figura 1.2: Autómata Finito Determinista del Ejercicio 1.2.1

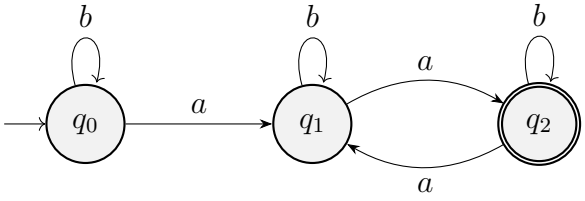


Figura 1.3: Autómata Finito Determinista del Ejercicio 1.2.2

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,

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

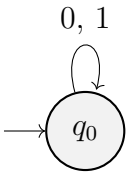
Ejercicio 1.2.2. Dado el AFD de la Figura 1.3, describir el lenguaje aceptado por dicho autómata.

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

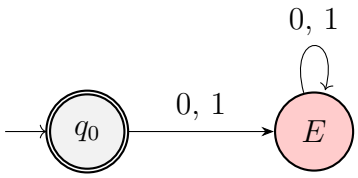
$$L = \{u \in \{a,b\}^* \mid n_a(u) \text{ es par}, \ n_a(u) > 0\},$$

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

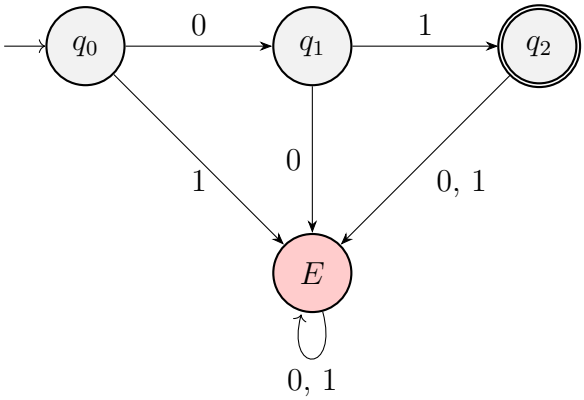
1. El lenguaje vacío,



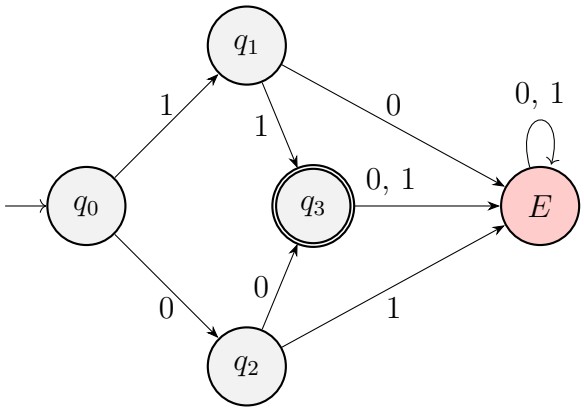
2. El lenguaje formado por la palabra vacía, es decir, $\{\varepsilon\}$,



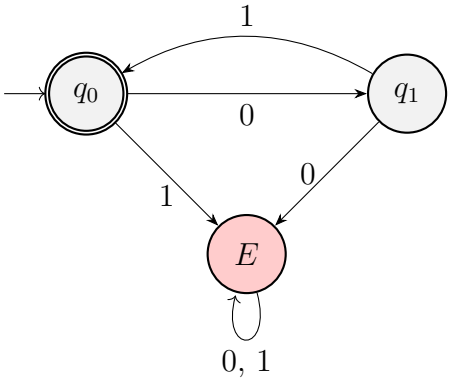
3. El lenguaje formado por la palabra 01, es decir, $\{01\}$,



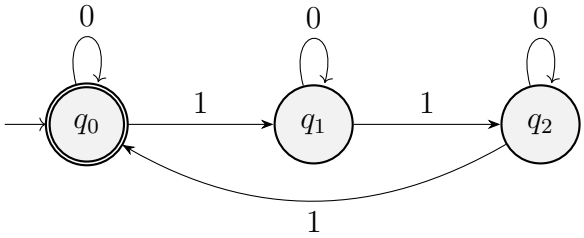
4. El lenguaje $\{11, 00\}$,



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.

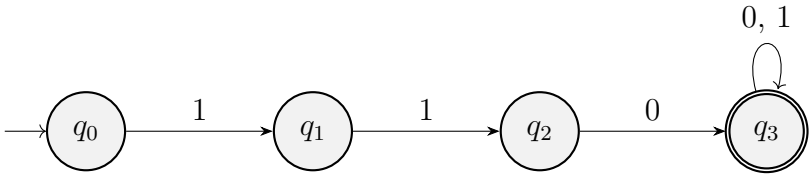


Figura 1.4: Autómata Finito No Determinista del Ejercicio 1.2.4

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.

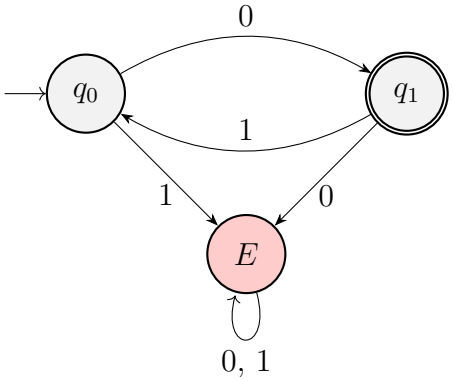


Figura 1.5: Autómata Finito Determinista del Ejercicio 1.2.5

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.

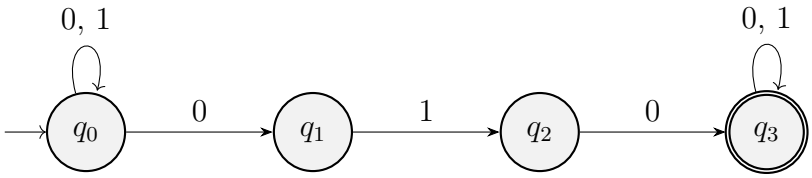


Figura 1.6: Autómata Finito No Determinista del Ejercicio 1.2.6 apartado 1.

2. AFND. Contengan la subcadena 110.

   El autómata obtenido es el de la Figura 1.7.

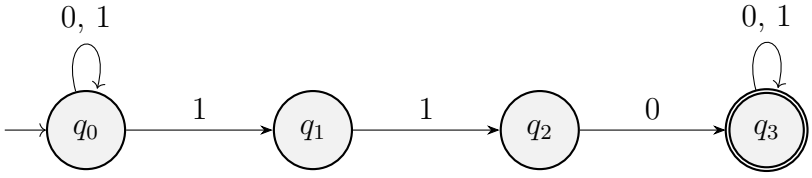


Figura 1.7: Autómata Finito No Determinista del Ejercicio 1.2.6 apartado 2.

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.

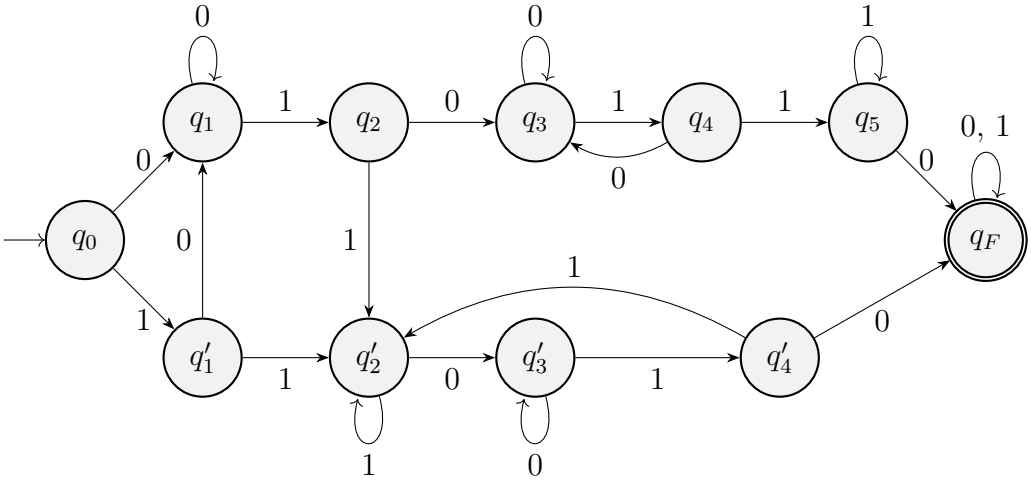


Figura 1.8: Autómata Finito Determinista del Ejercicio 1.2.6 apartado 3.

- 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.

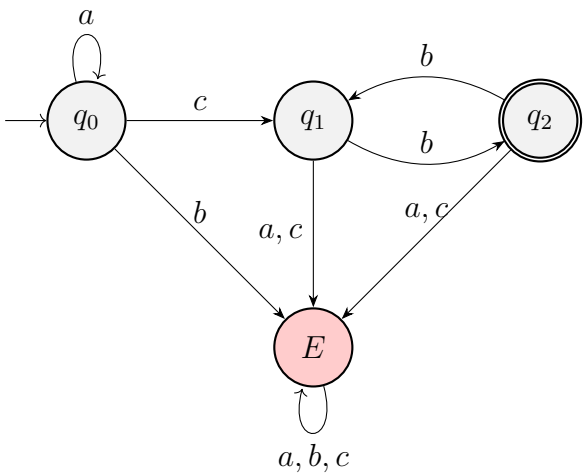


Figura 1.9: Autómata Finito Determinista del Ejercicio 1.2.7

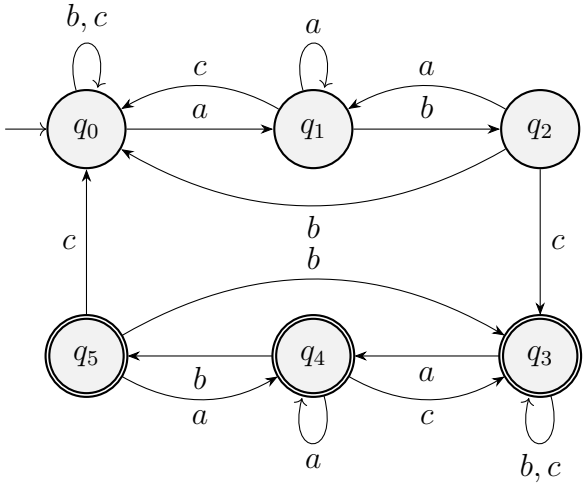


Figura 1.10: Autómata Finito Determinista del Ejercicio 1.2.8

- 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

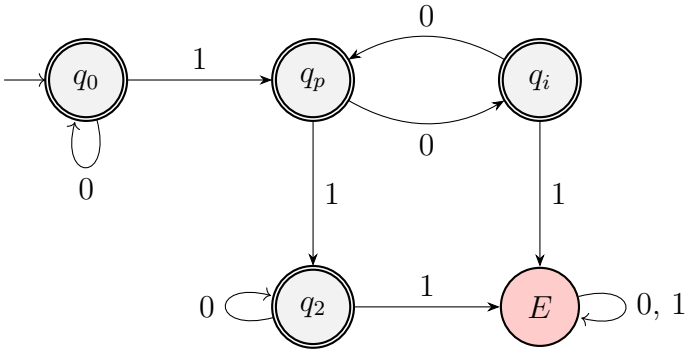


Figura 1.11: Autómata Finito Determinista del Ejercicio 1.2.9.

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

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

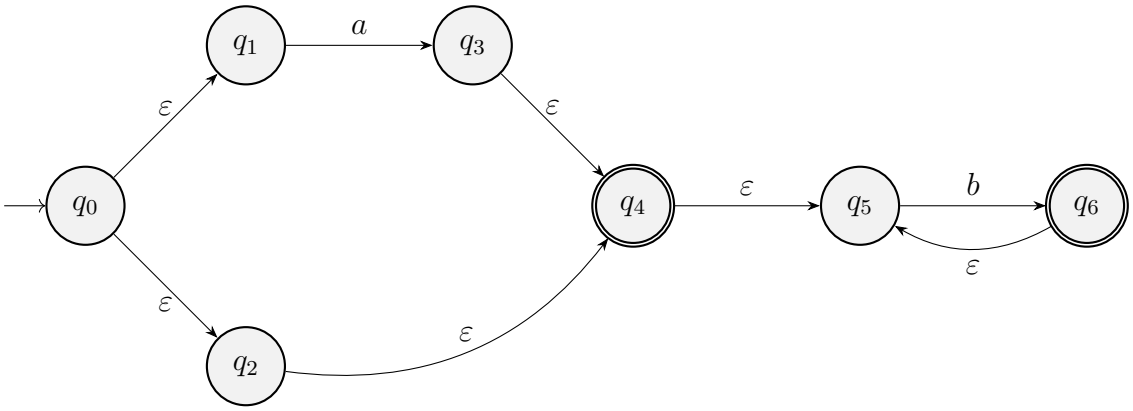


Figura 1.12: Autómata Finito No Determinista algorítmico del Ejercicio 1.2.10.

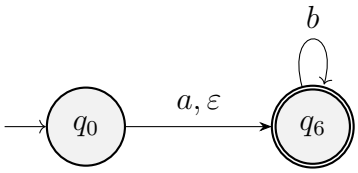


Figura 1.13: Autómata Finito No Determinista simplificado del Ejercicio 1.2.10.

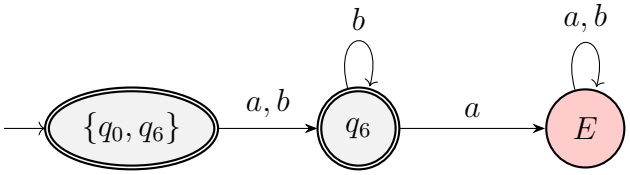


Figura 1.14: Autómata Finito Determinista del Ejercicio 1.2.10.

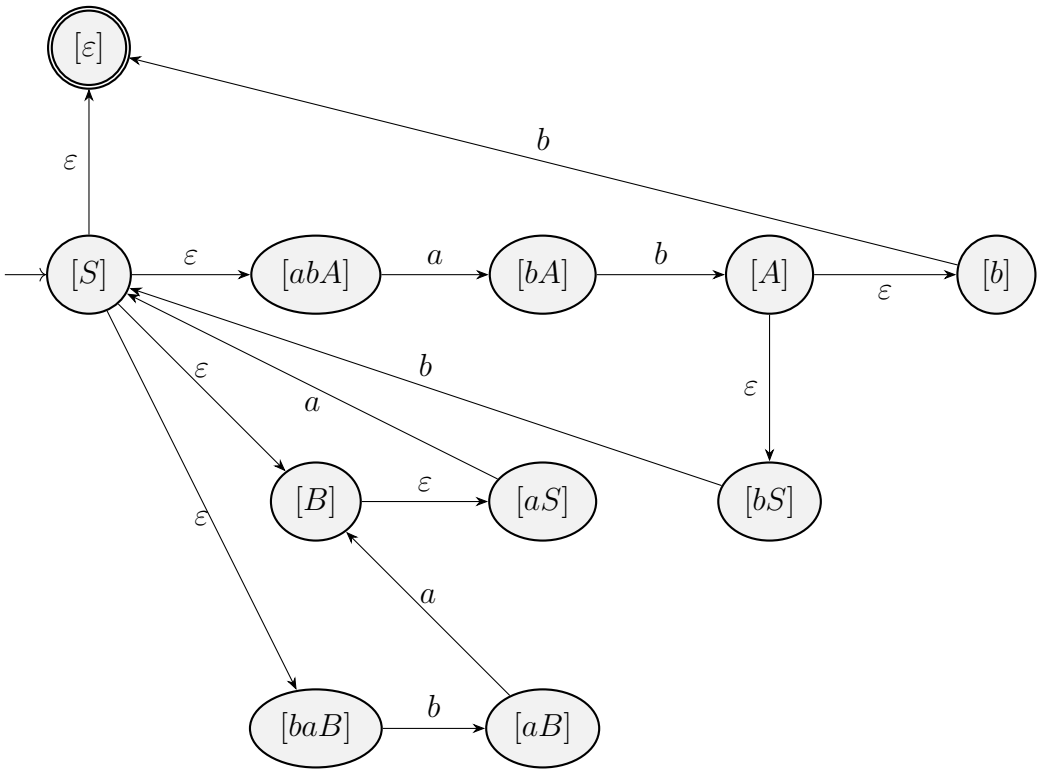


Figura 1.15: Autómata Finito Determinista del lenguaje $\mathcal{L}(S)$ del Ejercicio 1.2.11.

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:

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

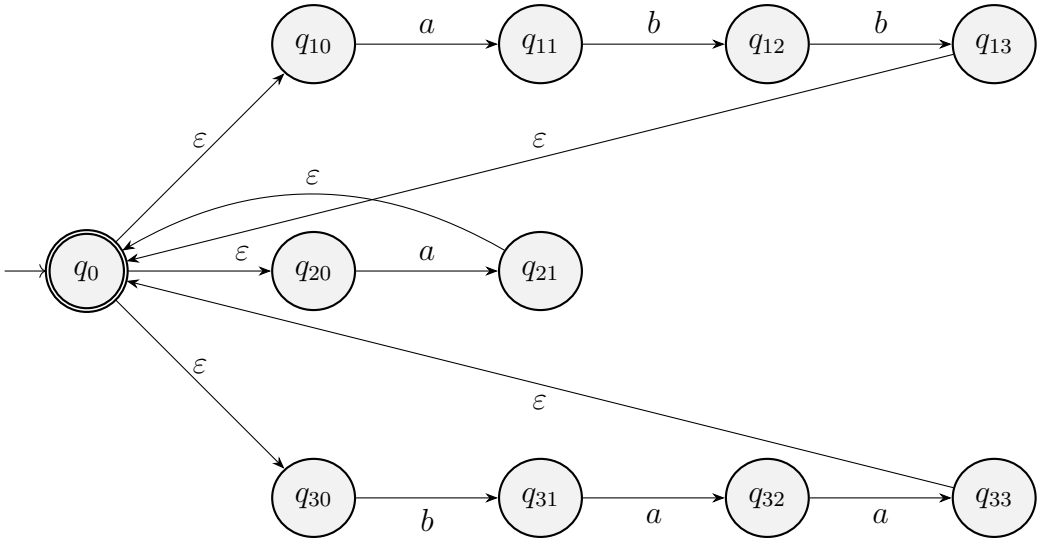


Figura 1.16: AFND asociado a $\mathcal{L}(G)$ del Ejercicio 1.2.11.

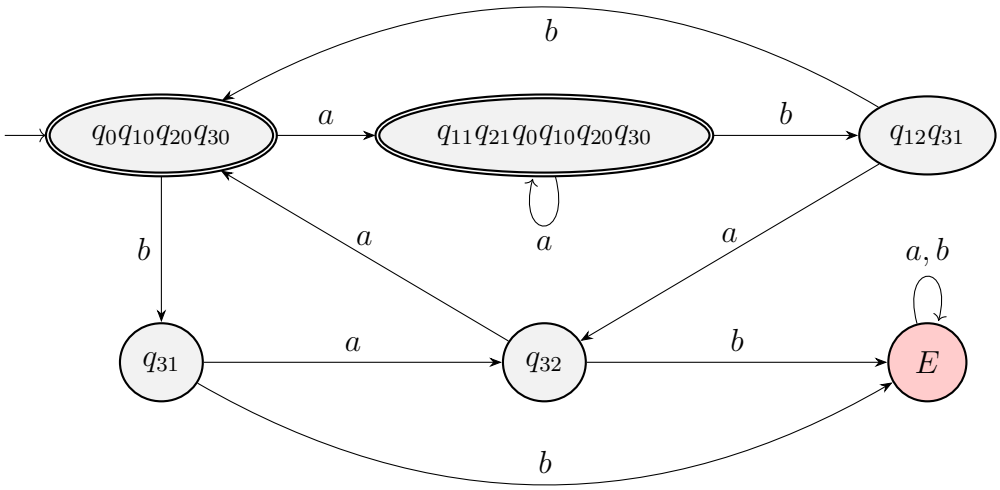


Figura 1.17: AFD asociado a $\mathcal{L}(G)$ del Ejercicio 1.2.11.

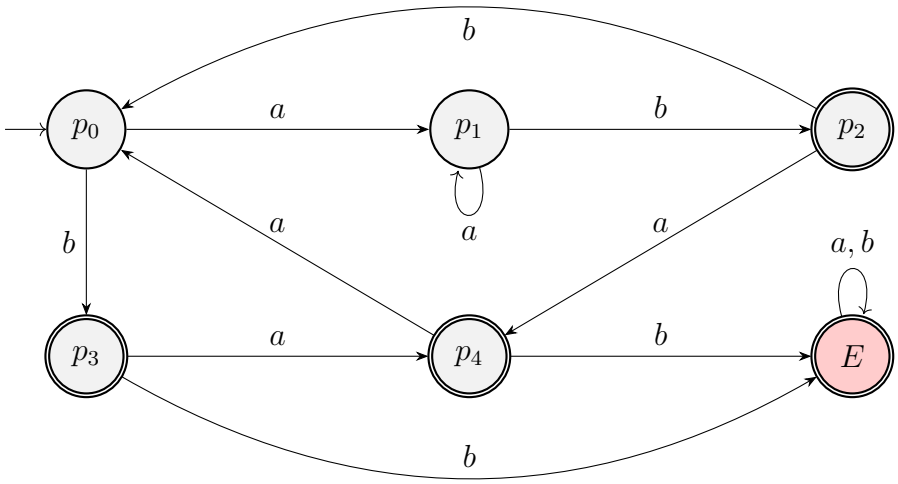


Figura 1.18: Autómata Finito Determinista del lenguaje $\overline{\mathcal{L}(G)}$ del Ejercicio 1.2.11.

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:

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

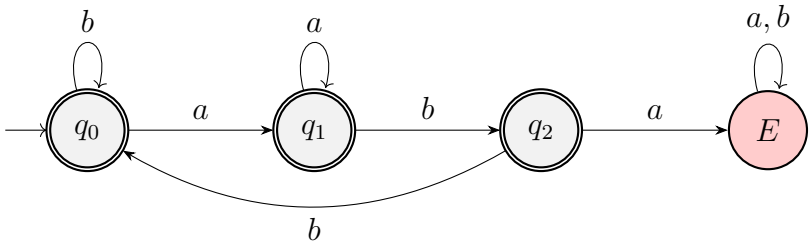


Figura 1.19: AFD del lenguaje del Ejercicio 1.2.12 apartado 3.

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.

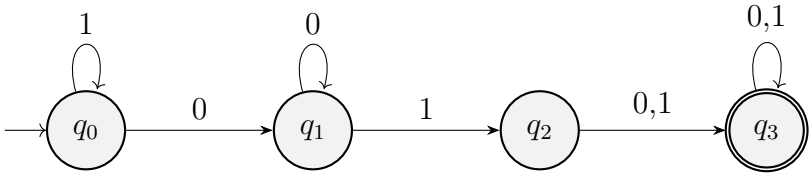


Figura 1.20: Autómata Finito Determinista del Ejercicio 1.2.14 apartado 1.

- $\underline{q_1}$: 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:

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

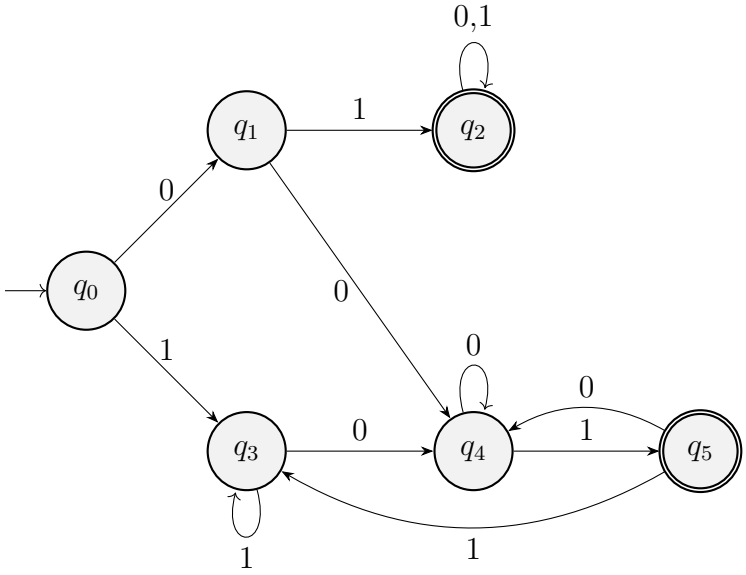


Figura 1.21: Autómata Finito Determinista del Ejercicio 1.2.14 apartado 2.

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:

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

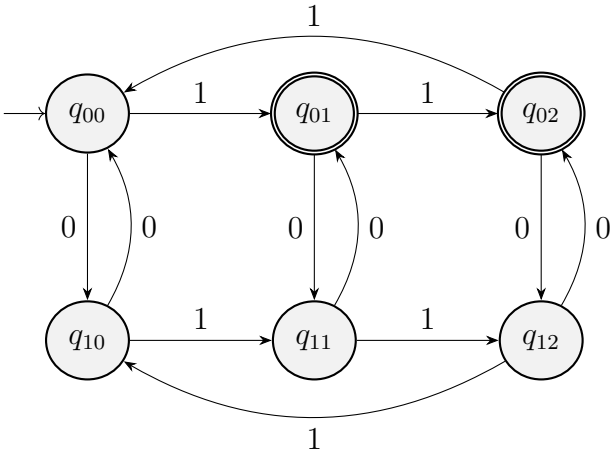


Figura 1.22: Autómata Finito Determinista del Ejercicio 1.2.17

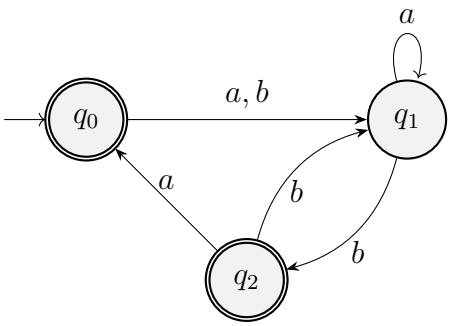


Figura 1.23: Autómata Finito Determinista del Ejercicio 1.2.18

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.

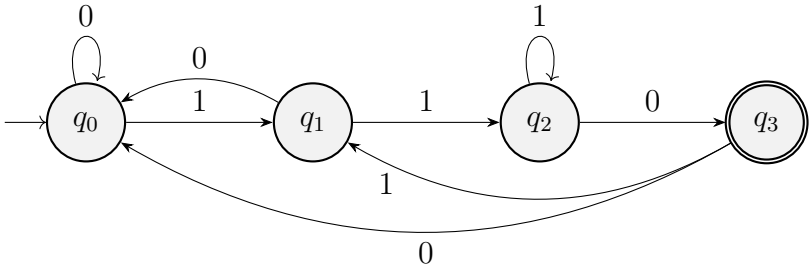


Figura 1.24: Autómata Finito Determinista del Ejercicio 1.2.19

- $\underline{q_1}$: 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.

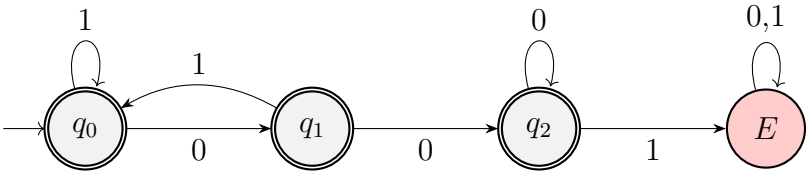


Figura 1.25: Autómata Finito Determinista del Ejercicio 1.2.21

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

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

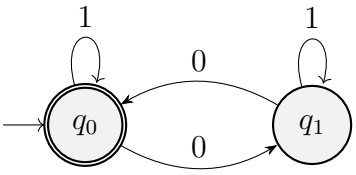
Dado un lenguaje L, se llama $\mathrm{Per}(L) = \bigcup_{u \in L} \mathrm{Per}(u)$. Dar expresiones regulares y autómatas minimales para $\mathrm{Per}(L)$ en los siguientes casos:

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:

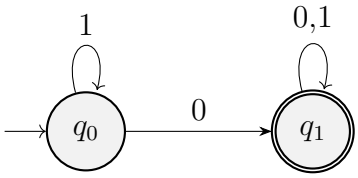
$$q_0 = 1q_0 + 0(1^*0q_0) + \varepsilon$$
$$q_0 = 1q_0 + 01^*0q_0 + \varepsilon$$
$$q_0 = (1 + 01^*0)q_0 + \varepsilon$$
$$q_0 = (1 + 01^*0)^*$$

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:

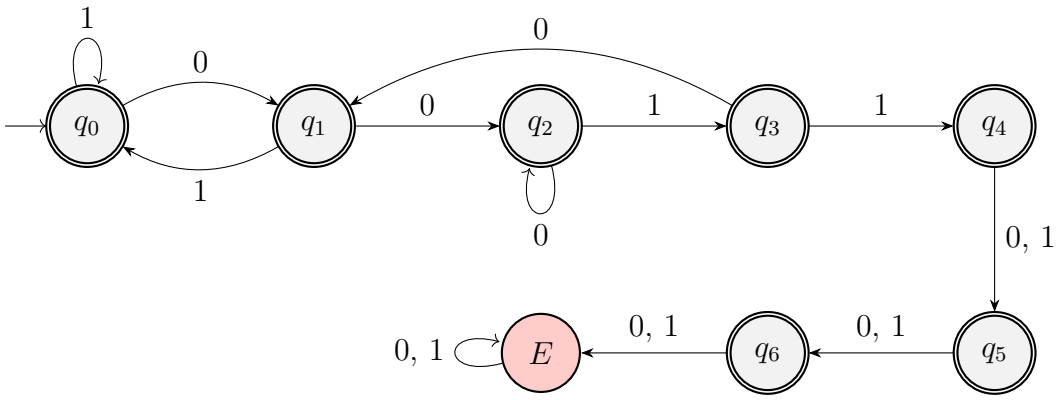


Figura 1.26: Autómata Finito Determinista para la pregunta 34.

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.