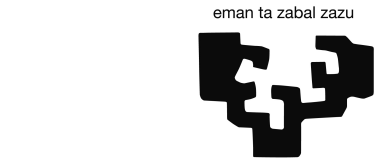


Lenguajes, Computación y Sistemas Inteligentes

Grado en Ingeniería Informática de Gestión y Sistemas de Información

Escuela de Ingeniería de Bilbao (UPV/EHU)

2^o curso

Curso académico 2023-2024

Tema 3: Lenguajes

JOSÉ GAINTZARAIN IBARMIA

Departamento de Lenguajes y Sistemas Informáticos

Última actualización: 04 - 09 - 2023

Índice general

Índice de figuras

Índice de tablas

Tema 3

Lenguajes

3.1.

Introducción

Para presentar los primeros resultados teóricos sobre computabilidad e incomputabilidad, introduciremos la teoría de los lenguajes formales. Un lenguaje formal es un conjunto formado por secuencias de símbolos, es decir, por palabras.

Cada especificación de un problema o cálculo, cada algoritmo y cada programa es una secuencia de símbolos. La teoría sobre lenguajes es la teoría sobre conjuntos formados por secuencias de símbolos. Dicha teoría permite probar que para algunas funciones o cálculos no es posible formular una especificación y también permite probar que para algunas funciones o cálculos no es posible diseñar un algoritmo (tampoco un programa).

En este tema, primero se presentarán los siguientes conceptos: símbolos, alfabetos, palabras, operaciones sobre palabras, conjunto de todas las palabras generables sobre un alfabeto, lenguajes (es decir, conjuntos de palabras), el conjunto de todos los lenguajes definibles sobre un alfabeto y, por último, operaciones sobre lenguajes.

Tras presentar los conceptos relacionados con las secuencias de símbolos, se estudiará la enumerabilidad de algunos conjuntos infinitos y la no enumerabilidad de otros conjuntos infinitos. En concreto, se demostrará que el conjunto de los números naturales, el conjunto formado por todos los pares de números naturales, el conjunto formado por todas las palabras y el conjuto formado por todos los pares de palabras son enumerables. Como consecuencia de estos resultados de enumerabilidad, tenemos que toda la información puede ser codificada o expresada utilizando números naturales. En cuanto a la no enumerabilidad de conjuntos infinitos, se demostrará que el conjunto formado por todos los lenguajes no es enumerable.

Para terminar el tema, se combinarán los resultados de enumerabilidad y no enumerabilidad y se probará que hay lenguajes no decidibles. Esto quiere decir que en el caso de algunos lenguajes, no es posible diseñar un algoritmo que, dada una palabra, sea capaz de decidir si esa palabra pertenece o no pertenece al lenguaje. Decidir significa responder “Sí” o “No”. Además, se probará también que hay lenguajes no semidecidibles. Un lenguaje es semidecidible si existe un algoritmo que, dada una palabra, realiza lo siguiente: si la palabra pertenece al lenguaje, el algoritmo responderá “Sí”; si la palabra no pertenece al lenguaje, el algoritmo no responderá,

es decir, se introducirá en un proceso de cálculo infinito.

En la tabla 3.8.1 (página 98), se recogen los conjuntos que se utilizan como tipos de datos en este tema.

3.2.

Alfabetos y palabras

En este apartado, se partirá del concepto de símbolo y primero se definirán los conceptos de alfabeto, palabra sobre un alfabeto y el conjunto de todas las palabras generables sobre un alfabeto. Para terminar se darán operaciones sobre palabras.

3.2.1 Símbolos

Se pueden utilizar como símbolos las letras, los dígitos, los símbolos de puntuación y demás elementos habituales en los juegos de caracteres disponibles en los ordenadores (flechas, caracteres especiales, etc).

3.2.2 Alfabetos

Un alfabeto es un conjunto **finito no vacío** de símbolos. A continuación se muestran algunos ejemplos:

$\mathbb{A}_1 = \{a, b, c, d, \ldots, z\}$ $\mathbb{A}_2 = \{a, b, c\}$ $\mathbb{A}_3 = \{x, y, z\}$

$\mathbb{A}_4 = \{0, 1\}$ $\mathbb{A}_5 = \{0, 1, 2, 3, \ldots, 9\}$ $\mathbb{A}_6 = \{1, \#, \sqcup\}$

$\mathbb{A}_7 = \{a, b, c, A, B, \#, \sqcup, @, \&\}$ $\mathbb{A}_8 = \{0, 1, \leftarrow, \uparrow, \rightarrow, \downarrow\}$ $\mathbb{A}_9 = \{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$

Para indicar que un símbolo α pertenece al alfabeto $\mathbb{A}$, escribiremos $\alpha \in \mathbb{A}$. Para indicar que un símbolo α no pertenece al alfabeto $\mathbb{A}$, escribiremos $\alpha \notin \mathbb{A}$.

3.2.3 Palabras

3.2.3.1 Definición de palabra

Dado un alfabeto $\mathbb{A}$, una **palabra** es una secuencia **finita** de símbolos que pertenecen al alfabeto $\mathbb{A}$, pudiendo repetirse los símbolos. La secuencia vacía o **palabra vacía** es representada

mediante el símbolo especial ε [1]. Pero ε no será un símbolo del alfabeto.

De manera formal, la noción de palabra sobre un alfabeto $\mathbb{A}$ se define recursivamente de la siguiente forma:

- La secuencia vacía ε es una palabra sobre el alfabeto $\mathbb{A}$.
- si α es un símbolo de $\mathbb{A}$ y w es una palabra sobre el alfabeto $\mathbb{A}$, entonces αw es una palabra sobre el alfabeto $\mathbb{A}$.
- Solo los elementos que se ajustan a los dos puntos anteriores son palabras sobre el alfabeto $\mathbb{A}$.

Por ejemplo, si consideramos el alfabeto $\mathbb{A}_2$ del apartado 3.2.2, las siguientes secuencias de caracteres son palabras definidas sobre el alfabeto $\mathbb{A}_2$:

$$aaaaa\varepsilon \qquad ba\varepsilon \qquad a\varepsilon \qquad bbbbabbbbaab\varepsilon$$

Esa es la representación abreviada de las estructuras lineales siguientes:

$$(a(a(a(a(a(a())))))) \qquad (b(a())) \qquad (a()) \qquad (b(b(b(b(a(b(b(b(b(a(a(b()))))))))))))$$

Para simplificar la escritura, se eliminan los paréntesis y en el caso de la estructura vacía $()$, para que su presencia quede patente, se pone ε.

Toda palabra puede ser generada a partir de la estructura vacía $()$ añadiendo símbolos de uno en uno por la izquierda. Así la palabra $(a(b(a(a(c()))))))$ sobre el alfabeto $\mathbb{A} = \{a, b, c\}$ se puede generar paso a paso de la siguiente forma:

$$() \Rightarrow (c()) \Rightarrow (a(c())) \Rightarrow (a(a(c()))) \Rightarrow (b(a(a(c()))))) \Rightarrow (a(b(a(a(c()))))))$$

En la versión abreviada o simplificada, el proceso se representaría de la siguiente manera:

$$\varepsilon \Rightarrow c\varepsilon \Rightarrow ac\varepsilon \Rightarrow aac\varepsilon \Rightarrow baac\varepsilon \Rightarrow abaac\varepsilon$$

Por tanto, para representar la palabra que se obtiene al añadir un símbolo α a una palabra v, hay que yuxtaponer α y v, es decir, hay que poner el símbolo α en cuestión seguido de la palabra v:

$$\alpha v$$

Dado un alfabeto $\mathbb{A}$, cada elemento α de $\mathbb{A}$ es un símbolo pero no una palabra. Por ejemplo, si $\mathbb{A} = \{a, b, c\}$, entonces a, b y c son símbolos de $\mathbb{A}$ pero no son palabras definidas sobre $\mathbb{A}$. En cambio, $a\varepsilon$, $b\varepsilon$ y $c\varepsilon$ son palabras sobre el alfabeto $\mathbb{A}$. Por tanto, a y $a\varepsilon$ son elementos de distinto tipo, son incomparables, de la misma forma que un entero no es comparable con un booleano y un entero no es comparable con una lista de enteros.

[1] ε: épsilon

Dado un alfabeto $\mathbb{A}$, denotaremos mediante $\mathbb{A}^*$ al **conjunto formado por todas las palabras** definidas sobre el alfabeto $\mathbb{A}$. Para indicar que una secuencia de símbolos w pertenece al conjunto $\mathbb{A}^*$ escribiremos $w \in \mathbb{A}^*$.

Formalmente, el conjunto $\mathbb{A}^*$ se puede definir de la siguiente manera:

- $\varepsilon \in \mathbb{A}^*$.
- Si $\alpha \in \mathbb{A}$ y $w \in \mathbb{A}^*$, entonces $\alpha w \in \mathbb{A}^*$.
- Solo los elementos que se ajustan a los dos puntos anteriores son elementos de $\mathbb{A}^*$.

Consecuentemente, tenemos que si $v \in \mathbb{A}^*$, entonces o $v = \varepsilon$ o $\exists \beta, u(\beta \in \mathbb{A} \wedge u \in \mathbb{A}^* \wedge v = \beta u)$.

Por ejemplo, si $\mathbb{A} = \{a, b\}$, entonces

$$\mathbb{A}^* = \{\varepsilon, a\varepsilon, b\varepsilon, aa\varepsilon, ab\varepsilon, ba\varepsilon, bb\varepsilon, aaa\varepsilon, aab\varepsilon, aba\varepsilon, ab\varepsilon, baa\varepsilon, bab\varepsilon, bba\varepsilon, bbb\varepsilon, aaaa\varepsilon, \dots\}$$

Por otro lado, si consideramos que v es la palabra $aba\varepsilon$ de $\mathbb{A}^*$, en esa palabra β será a y u será $ba\varepsilon$: $\underbrace{a}_{\beta}\underbrace{ba\varepsilon}_{u}$

Si consideramos que v es la palabra $a\varepsilon$ de $\mathbb{A}^*$, en esa palabra β será a y u será ε: $\underbrace{a}_{\beta}\underbrace{\varepsilon}_{u}$

El conjunto $\mathbb{A}^*$ es **infinito** para cualquier alfabeto $\mathbb{A}$.

3.2.3.2 Definición de subpalabra

Sea w una palabra definida sobre un alfabeto $\mathbb{A}$. Cualquier secuencia de símbolos que aparezcan en posiciones consecutivas en w, es una subpalabra de w.

Por ejemplo, consideremos la palabra $w = ccbacbbaa\varepsilon$ definida sobre el alfabeto $\mathbb{A} = \{a, b, c\}$. Las palabras ε, $ccba\varepsilon$, $acb\varepsilon$, $ccbacbbaa\varepsilon$, $ac\varepsilon$ y $c\varepsilon$ son subpalabras de w. Por otra parte, las palabras $acc\varepsilon$, $bbb\varepsilon$ y $aaaa\varepsilon$ no son subpalabras de w.

ε es subpalabra de cualquier palabra. Dada una palabra cualquiera w, esta será subpalabra de ella misma (w es subpalabra de w).

3.2.4 Operaciones sobre palabras

A continuación, se definen nueve operaciones sobre palabras.

3.2.4.1 Longitud de una palabra

Sea w una palabra definida sobre un alfabeto $\mathbb{A}$. La **longitud** de w es el número de símbolos que forman la palabra y se denota como $|w|$. El tipo de la función que calcula la longitud es el siguiente:

$$\mathbb{A}^* \to \mathbb{N}$$

Mediante el tipo se indica que, dado un elemento del conjunto $\mathbb{A}^*$ (es decir, dada una palabra), se obtiene un número natural.[2]

Por ejemplo, considerando las palabras $aaaaa\varepsilon$, $ba\varepsilon$, $a\varepsilon$ y $bbbbabbbbaab\varepsilon$ definidas sobre el alfabeto $\mathbb{A} = \{a, b\}$, tenemos que $|aaaaa\varepsilon| = 5$, $|ba\varepsilon| = 2$, $|a\varepsilon| = 1$ y $|bbbbabbbbaab\varepsilon| = 12$. El símbolo especial ε no se cuenta. El símbolo especial ε se pone únicamente para indicar que tenemos una palabra. La longitud de la palabra vacía es 0, es decir, $|\varepsilon| = 0$.

La definición formal de la longitud es la siguiente:

- $|\varepsilon| = 0$
- Si $\alpha \in \mathbb{A}$ y $w \in \mathbb{A}^*$, entonces $|\alpha w| = 1 + |w|$.

3.2.4.2 Número de apariciones de un símbolo en una palabra

Sea w una palabra definida sobre un alfabeto $\mathbb{A}$ y α un símbolo de $\mathbb{A}$. Denotaremos mediante la expresión $|w|_\alpha$ el número de apariciones del símbolo α en la palabra w. El tipo de la función es el siguiente:

$$\mathbb{A}^* \times \mathbb{A} \to \mathbb{N}$$

Mediante el tipo se indica que, dado un elemento del conjunto $\mathbb{A}^*$ (es decir, dada una palabra) y un símbolo de $\mathbb{A}$, se obtiene un número natural.

Por ejemplo, si consideramos las palabras $ba\varepsilon$, $a\varepsilon$, $bbbbabbbbaab\varepsilon$ y ε definidas sobre el alfabeto $\mathbb{A} = \{a, b\}$, tenemos que $|ba\varepsilon|_b = 1$, $|a\varepsilon|_b = 0$, $|bbbbabbbbaab\varepsilon|_a = 3$, $|bbbbabbbbaab\varepsilon|_b = 9$, $|\varepsilon|_b = 0$.

Nótese que ε no es un símbolo de $\mathbb{A}$ y, por tanto, es incorrecto preguntar por el número de apariciones de ε.

La definición formal es la siguiente:

- $|\varepsilon|_\alpha = 0$, para cualquier símbolo α perteneciente a $\mathbb{A}$
- Si $\alpha \in \mathbb{A}$, $\beta \in A$ y $w \in \mathbb{A}^*$, entonces $|\beta w|_\alpha = \begin{cases} 1 + |w|_\alpha & \text{si } \alpha = \beta \\ |w|_\alpha & \text{si } \alpha \neq \beta \end{cases}$

[2] Mediante $\mathbb{N}$ denotamos el conjunto infinito de los números naturales $\{0, 1, 2, 3, \dots\}$

3.2.4.3 Símbolo de una determinada posición

Sea w una palabra no vacía definida sobre un alfabeto $\mathbb{A}$ y sea k un número entero comprendido entre 1 y $|w|$, es decir, $1 \leq k \leq |w|$. Denotaremos mediante $w(k)$ el símbolo que ocupa la posición k empezando desde la izquierda. Se considera que la posición del extremo izquierdo es 1. La posición 0 no existe. El tipo de la función es el siguiente:

$$\mathbb{A}^* \times \mathbb{N} \to \mathbb{A}$$

Mediante el tipo se indica que, dado un elemento del conjunto $\mathbb{A}^*$ (es decir, dada una palabra) y un número natural, se obtiene un símbolo del alfabeto $\mathbb{A}$.

Por ejemplo, considerando la palabra $bbbbaacb\varepsilon$ definida sobre el alfabeto $\mathbb{A} = \{a, b, c\}$, tenemos que $bbbbaacb\varepsilon(2) = b$, $bbbbaacb\varepsilon(5) = a$ y $bbbbaacb\varepsilon(7) = c$.

No es correcto plantear $bbbbaacb\varepsilon(9)$ ni $bbbbaacb\varepsilon(20)$ ni $bbbbaacb\varepsilon(0)$ ni $bbbbaacb\varepsilon(-3)$. En todos estos casos, el número aportado no está entre 1 y la longitud de la palabra (la longitud es 8).

Definición formal:
Sean $\alpha \in \mathbb{A}$, $v \in \mathbb{A}^*$ eta $k \in \{1, \ldots, |\alpha v|\}$:

$$\alpha v(k) = \begin{cases} \alpha & \text{si } k = 1 \\ v(k-1) & \text{si } k \neq 1 \end{cases}$$

La operación está definida solo para palabras no vacías, es decir, palabras que tienen la forma αv siendo α un símbolo de $\mathbb{A}$ y v una palabra definida sobre $\mathbb{A}$.

3.2.4.4 Subcadena comprendida entre dos posiciones

Sea w una palabra definida sobre un alfabeto $\mathbb{A}$, sea k un número natural comprendido entre 1 y $|w| + 1$ y sea ℓ un número natural comprendido entre 0 y $|w|$. Denotaremos mediante $w(k, \ell)$ la subpalabra de w formada por los símbolos que van desde la posición k hasta la posición ℓ. Las posiciones k y ℓ también se incluyen. Si k es mayor que ℓ, se devuelve la palabra vacía ε. Consecuentemente, si k es $|w| + 1$ o ℓ es 0, se devuelve la palabra vacía ε. Las posiciones se contabilizan empezando desde la izquierda. La primera posición es la posición 1. La posición 0 no existe. El tipo de la función es el siguiente:

$$\mathbb{A}^* \times \mathbb{N} \times \mathbb{N} \to \mathbb{A}^*$$

Mediante el tipo se indica que, dado un elemento del conjunto $\mathbb{A}^*$ (es decir, dada una palabra) y dos números naturales, se obtiene un elemento del conjunto $\mathbb{A}^*$ (es decir, una palabra).

Por ejemplo, considerando la palabra $bbcbaacb\varepsilon$ definida sobre el alfabeto $\mathbb{A} = \{a, b, c\}$, tenemos que $bbcbaacb\varepsilon(3, 6) = cbaa\varepsilon$, $cbaacb\varepsilon(5, 5) = c\varepsilon$, $cbaacb\varepsilon(7, 6) = \varepsilon$, $cbaacb\varepsilon(1, 0) = \varepsilon$, $bbbbaacb\varepsilon(5, 4) = \varepsilon$ y $bbbbaacb\varepsilon(5, 2) = \varepsilon$.

Definición formal:
Sean $w \in \mathbb{A}^*$, $k \in \{1, \ldots, |w| + 1\}$ y $\ell \in \{0, \ldots, |w|\}$:

$$w(k, \ell) = \begin{cases} \varepsilon & \text{si } k > \ell \\ (w(k))(w(k+1, \ell)) & \text{si } k \leq \ell \end{cases}$$

Nótese que si w es ε, entonces k ha de ser necesariamente 1 y ℓ ha de ser necesariamente 0. El resultado de $\varepsilon(1, 0)$ es ε porque $k > \ell$, es decir, $1 > 0$.

3.2.4.5 Concatenación de dos palabras

Dadas dos palabras w y v definidas sobre un alfabeto $\mathbb{A}$, la palabra z que se obtiene al concatenarlas es la palabra $w \cdot v$. El operador de la operación de concatenación de palabras es $\cdot$.

El tipo de la función de concatenación es el siguiente:

$$\mathbb{A}^* \times \mathbb{A}^* \to \mathbb{A}^*$$

Mediante el tipo se indica que, dados dos elementos del conjunto $\mathbb{A}^*$ (es decir, dadas dos palabras), se obtiene un elemento del conjunto $\mathbb{A}^*$ (es decir, otra palabra).

Definición formal:

- $\varepsilon \cdot v = v$ para cualquier palabra v perteneciente a $\mathbb{A}^*$.
- Siendo α un símbolo cualquiera del alfabeto $\mathbb{A}$ y siendo u y v dos palabras cualesquiera de A^*, la palabra que se obtiene concatenando las palabras αu y v es la palabra generada al añadir el símbolo α a la palabra formada al concatenar las palabras u y v. Es decir, $(\alpha u) \cdot v = \alpha(u \cdot v)$.

La palabra z obtenida al concatenar las palabras w y v tiene las siguientes **características**:

- Si $|w| = m$ y $|v| = n$, entonces $|z| = m + n$
- Si $|w| = m$ y $1 \leq j \leq m$, entonces $z(j) = w(j)$
- Si $|w| = m$, $|v| = n$ y $m + 1 \leq j \leq m + n$, entonces $z(j) = v(j - m)$

Por ejemplo, sean $w = aa\varepsilon$ y $v = ba\varepsilon$ dos palabras definidas sobre el alfabeto $\mathbb{A} = \{a, b\}$. La palabra $w \cdot v$ es $aaba\varepsilon$, mientras que la palabra $v \cdot w$ es $baaa\varepsilon$.

La concatenación es **asociativa**, es decir, si u, v y w son tres palabras definidas sobre un mismo alfabeto, se cumple $u \cdot (v \cdot w) = (u \cdot v) \cdot w$. Por tanto, no es necesario utilizar paréntesis. Esto significa que, por ejemplo, al expresar la concatenación de cuatro palabras u, v, w y x, no hace falta poner paréntesis: $u \cdot v \cdot w \cdot x$.

El **elemento neutro** de la concatenación es la palabra vacía ε. Consecuentemente, $\varepsilon \cdot w = w = w \cdot \varepsilon$ para cualquier palabra w. En particular, $\varepsilon \cdot \varepsilon = \varepsilon$.

También se cumple, $w(1,k) \cdot w(k+1,|w|) = w$, donde w es cualquier palabra y $0 \leq k \leq |w|$. Esta propiedad establece una relación entre la operación que devuelve la subpalabra comprendida entre dos posiciones y la operación que concatena dos palabras.

3.2.4.6 Exponenciación de palabras

Dados una palabra w definida sobre un alfabeto $\mathbb{A}$ y un número natural j, la palabra z que se obtiene al elevar w a j, y que denotaremos como w^j, es la palabra que se obtiene al concatenar w j veces: $\underbrace{w \cdot w \cdot \ ... \ \cdot w}_{j \text{ veces}}$.

El tipo de la función de exponenciación es el siguiente:

$$\mathbb{A}^* \times \mathbb{N} \rightarrow \mathbb{A}^*$$

Mediante el tipo se indica que, dados un elemento del conjunto $\mathbb{A}^*$ (es decir, dada una palabra) y un número natural, se obtiene un elemento del conjunto $\mathbb{A}^*$ (es decir, otra palabra).

Definición formal:

- Para cualquier palabra w perteneciente a $\mathbb{A}^*$, se cumple $w^0 = \varepsilon$.
- Para cualquier palabra w perteneciente a $\mathbb{A}^*$ y cualquier número natural j que sea ≥ 1, se cumple $w^j = w \cdot (w^{j-1})$.

La palabra z obtenida al calcular w^j tiene la siguiente **característica**: Si $|w| = n$, entonces $|z| = n * j$ (aquí $*$ representa la multiplicación o producto entre números). Además, se cumple $w^j \cdot w^k = w^{j+k}$, $w^1 = w$ y $w^0 = \varepsilon$.

Por ejemplo, sean $w = baa\varepsilon$ una palabra definida sobre el alfabeto $\mathbb{A} = \{a, b\}$ y $j = 3$. La palabra w^j es $baa\varepsilon \cdot baa\varepsilon \cdot baa\varepsilon$ donde w aparece 3 veces. Es decir, $w^3 = baabaabaa\varepsilon$. Recuérdese que al aplicar la definición de la operación de concatenación, las ε intermedias desaparecen y solo queda la del extremo derecho.

3.2.4.7 Inversa de una palabra

Dada una palabra w definida sobre un alfabeto $\mathbb{A}$, la palabra inversa de w es la palabra cuyos componentes están en orden inverso a los componentes de w. Denotaremos la inversa de w como w^R, donde R proviene de la palabra inglesa "Reverse".

El tipo de la función que calcula la inversa es el siguiente:

$$\mathbb{A}^* \rightarrow \mathbb{A}^*$$

Mediante el tipo se indica que, dado un elemento del conjunto $\mathbb{A}^*$ (es decir, dada una palabra), se obtiene un elemento del conjunto $\mathbb{A}^*$ (es decir, otra palabra).

Definición formal:

- $\varepsilon^R = \varepsilon$
- Para cualquier símbolo α perteneciente al alfabeto $\mathbb{A}$ y cualquier palabra w perteneciente al conjunto $\mathbb{A}^*$, se tiene que $(\alpha w)^R = (w^R) \cdot (\alpha\varepsilon)$.

Para colocar el símbolo α al final (en el extremo derecho) es necesario crear la palabra $\alpha\varepsilon$ y concatenar las palabras w^R y $\alpha\varepsilon$. No es posible concatenar directamente la palabra w^R y el símbolo α porque la concatenación ha de juntar dos palabras y no una palabra y un símbolo.

La palabra w^R obtenida al calcular la inversa de w tiene las siguientes **características**:

- $|w^R| = |w|$
- $\forall k(1 \leq k \leq |w| \rightarrow w^R(k) = w(|w| + 1 - k)$
- $(w^R)^R = w$.

Por ejemplo, sea $w = baa\varepsilon$ una palabra definida sobre el alfabeto $\mathbb{A} = \{a, b\}$. La palabra w^R es $aab\varepsilon$.

Nótese que ε siempre permanece en el extremo derecho.

En particular, si α es un símbolo del alfabeto $\mathbb{A}$, se cumple $(\alpha\varepsilon)^R = \alpha\varepsilon$. Por ejemplo, si consideramos el alfabeto $\mathbb{A} = \{a, b\}$, tenemos que $(a\varepsilon)^R = a\varepsilon$ y $(b\varepsilon)^R = (b\varepsilon)$. Además, para la palabra vacía tenemos que $\varepsilon^R = \varepsilon$.

3.2.4.8 Decidir si una palabra es prefijo de otra

Dadas dos palabras v y w definidas sobre un alfabeto $\mathbb{A}$, la palabra v será prefijo de la palabra w, si los símbolos que conforman la palabra v aparecen al principio de w en el mismo orden. Esta operación se denomina $es\_prefijo$.

El tipo de esta operación es el siguiente:

$$\mathbb{A}^* \times \mathbb{A}^* \rightarrow \mathbb{2}$$

Mediante el tipo se expresa que, dados dos elementos del conjunto $\mathbb{A}^*$ (es decir, dos palabras), se devolverá un elemento del conjunto $\mathbb{2}$.

El conjunto $\mathbb{2}$ es el conjunto $\{0, 1\}$, donde 0 representa *falso* y 1 representa *cierto*. Por tanto, el conjunto o tipo $\mathbb{2}$ viene a ser el tipo de los Booleanos.

Definición formal:

- Para cualquier palabra w de $\mathbb{A}^*$: $es\_prefijo(\varepsilon, w) = 1$.
- Para cualquier símbolo α de $\mathbb{A}$ y cualquier palabra x de $\mathbb{A}^*$: $es\_prefijo(\alpha x, \varepsilon) = 0$
- Para cualquier par de símbolos α y β de $\mathbb{A}$ y cualquier par de palabras w y x de $\mathbb{A}^*$:

$$es\_prefijo(\alpha x, \beta w) = \begin{cases} 0 & \text{si } \alpha \neq \beta \\ es\_prefijo(x, w) & \text{si } \alpha = \beta \end{cases}$$

La palabra vacía ε es prefijo de cualquier palabra.

Por ejemplo, si consideramos las palabras $baa\varepsilon$, $aa\varepsilon$ y $baaaaab\varepsilon$ definidas sobre el alfabeto $\mathbb{A} = \{a, b, c\}$, se cumple lo siguiente:

$$es\_prefijo(baa\varepsilon, baaaaab\varepsilon) = 1$$
$$es\_prefijo(aa\varepsilon, baaaaab\varepsilon) = 0$$

La operación $es\_prefijo$ cumple las siguientes **propiedades**:

$$(es\_prefijo(v, w) = 1) \leftrightarrow (\exists j(j \in \mathbb{N} \wedge (0 \leq j \leq |w|) \wedge v = w(1, j)))$$

$$(es\_prefijo(v, w) = 1) \leftrightarrow (\exists x(x \in \mathbb{A}^* \wedge w = v \cdot x))$$

3.2.4.9 Decidir si una palabra es subpalabra de otra

Dadas dos palabras v y w definidas sobre un alfabeto $\mathbb{A}$, la palabra v es subpalabra de la palabra w, si los símbolos que conforman la palabra v aparecen seguidos en w y en el mismo orden. Esta operación se denomina $es\_subpalabra$.

El tipo de esta operación es el siguiente:

$$\mathbb{A}^* \times \mathbb{A}^* \rightarrow \mathbb{2}$$

Mediante el tipo se expresa que, dados dos elementos del conjunto $\mathbb{A}^*$ (es decir, dos palabras), se devolverá un elemento del conjunto $\mathbb{2}$. Por tanto, se devolverá un 0 (falso) o un 1 (cierto).

Definición formal:

- Para cualquier palabra w de $\mathbb{A}^*$: $es\_subpalabra(\varepsilon, w) = 1$.
- Para cualquier símbolo α de $\mathbb{A}$ y cualquier palabra x de $\mathbb{A}^*$: $es\_subpalabra(\alpha x, \varepsilon) = 0$
- Para cualquier par de símbolos α y β de $\mathbb{A}$ y cualquier par de palabras w y x de $\mathbb{A}^*$:

$$es\_subpalabra(\alpha x, \beta w) = \begin{cases} 1 & \text{si } es\_prefijo(\alpha x, \beta w) = 1 \\ es\_subpalabra(\alpha x, w) & \text{en caso contrario} \end{cases}$$

La palabra vacía ε es subpalabra de cualquier palabra.

Por ejemplo, si consideramos las palabras $baa\varepsilon$, $aa\varepsilon$, $aba\varepsilon$, $cccc\varepsilon$ y $baaaaab\varepsilon$ definidas sobre el alfabeto $\mathbb{A} = \{a, b, c\}$, se cumple lo siguiente:

$$es\_subpalabra(baa\varepsilon, baaaaab\varepsilon) = 1$$
$$es\_subpalabra(aa\varepsilon, baaaaab\varepsilon) = 1$$
$$es\_subpalabra(aba\varepsilon, baaaaab\varepsilon) = 0$$
$$es\_subpalabra(cccc\varepsilon, baaaaab\varepsilon) = 0$$

La operación $es\_subpalabra$ cumple las siguientes **propiedades**:

$$(es\_subpalabra(v, w) = 1) \quad \leftrightarrow \quad (\exists j, k(j \in \mathbb{N} \wedge k \in \mathbb{N} \wedge (1 \leq j \leq |w| + 1) \wedge (0 \leq k \leq |w|) \wedge v = w(j, k)))$$

$$(es\_subpalabra(v, w) = 1) \quad \leftrightarrow \quad (\exists x, z(x \in \mathbb{A}^* \wedge z \in \mathbb{A}^* \wedge w = x \cdot v \cdot z))$$

3.3.

Lenguajes: definición y operaciones

3.3.1 Definición de lenguaje y lenguaje universal

En la sección 3.2.3 se ha dicho que el conjunto $\mathbb{A}^*$ está formado por todas las palabras finitas definibles sobre el alfabeto $\mathbb{A}$ y que es un conjunto infinito para cualquier alfabeto.

Cualquier subconjunto (finito o infinito, vacío o no vacío) de $\mathbb{A}^*$ es un **lenguaje definido sobre el alfabeto** $\mathbb{A}$. El propio $\mathbb{A}^*$ es también un lenguaje definido sobre el alfabeto $\mathbb{A}$. El conjunto $\mathbb{A}^*$ recibe el nombre de **lenguaje universal definido sobre** $\mathbb{A}$. Todos los demás lenguajes definidos sobre $\mathbb{A}$ son subconjuntos de $\mathbb{A}^*$. Denotamos el conjunto vacío mediante el símbolo especial $\varnothing$. Para expresar que una palabra w pertenece a un lenguaje L, escribimos $w \in L$ y para expresar que una palabra w no pertenece a un lenguaje L, escribimos $w \notin L$.

Por ejemplo, si consideramos el alfabeto $\mathbb{A} = \{a, b\}$, los siguientes conjuntos L_1, L_2, L_3, L_4 y L_5 son lenguajes definidos sobre $\mathbb{A}$:

$$
\begin{aligned}
L_1 &= \{aa\varepsilon, bab\varepsilon, bbb\varepsilon, aaaa\varepsilon\} \\
L_2 &= \{\varepsilon, a\varepsilon, aa\varepsilon, aaa\varepsilon, aaaa\varepsilon, \dots\} \\
L_3 &= \varnothing \\
L_4 &= \{\varepsilon\} \\
L_5 &= \{a\varepsilon, b\varepsilon\}
\end{aligned}
$$

El lenguaje L_2 es infinito y está formado por todas las palabras que se obtienen con cero o más repeticiones del símbolo a. Los otros cuatro lenguajes son finitos.

3.3.2 Un lenguaje representa la especificación de una tarea a informatizar

Dado un alfabeto $\mathbb{A}$, una palabra definida sobre $\mathbb{A}$ representa una estructura lineal formada por una combinación —con posibles repeticiones— de los elementos de $\mathbb{A}$. Por su parte, un lenguaje L definido sobre $\mathbb{A}$, está formado por un conjunto de palabras que, en general, cumplen

una determinada propiedad que las caracteriza con respecto al resto de elementos de $\mathbb{A}^*$ que no pertenecen a L.

En este contexto, un lenguaje L definido sobre $\mathbb{A}$, representa una tarea o un cálculo que se desea automatizar o informatizar. En concreto, la tarea que se desea informatizar es la que consiste en averiguar si un elemento w perteneciente a $\mathbb{A}^*$, cumple la propiedad que caracteriza a L. Por tanto, dada una palabra $w \in \mathbb{A}^*$, se desea saber si w pertenece a L o no.

Consecuentemente, cuando se da la definición de un lenguaje, realmente se da la **especificación** de la tarea que se pretende automatizar. Es decir, L representa la especificación de la tarea que se desea informatizar.

3.3.3 Representación formal de lenguajes como conjuntos

A la hora de definir formalmente un lenguaje, tenemos que considerar dos casos:

- Si el lenguaje es finito podemos definirlo poniendo todos sus componentes o, alternativamente, podemos definirlo dando una propiedad que sirva para diferenciar sus palabras del resto de las palabras de $\mathbb{A}^*$.
- Si el lenguaje es infinito tenemos que definirlo dando una propiedad que sirva para diferenciar sus palabras del resto de las palabras de $\mathbb{A}^*$.

Para expresar propiedades que sirven para definir lenguajes, se pueden utilizar las operaciones definidas sobre palabras y los elementos propios de la lógica de primer orden o lógica de predicados:

- $\neg$: "no", con formato $\neg\psi$, donde ψ es una propiedad o una fórmula de la lógica de primer orden.
- $\wedge$: "y", con formato $\psi_1 \wedge \psi_2$, donde ψ_1 y ψ_2 son dos propiedades o fórmulas de la lógica de primer orden.
- $\vee$: "o", con formato $\psi_1 \vee \psi_2$, donde ψ_1 y ψ_2 son dos propiedades o fórmulas de la lógica de primer orden.
- $\rightarrow$: "condicional" o "implicación", con formato $\psi_1 \rightarrow \psi_2$, donde ψ_1 y ψ_2 son dos propiedades o fórmulas de la lógica de primer orden.
- $\leftrightarrow$: "equivalencia", con formato $\psi_1 \leftrightarrow \psi_2$, donde ψ_1 y ψ_2 son dos propiedades o fórmulas de la lógica de primer orden.
- $\forall$: "cuantificador universal", con formato

  $$\forall x_1, x_2, \ldots, x_k (D(x_1, x_2, \ldots, x_k) \rightarrow P(x_1, x_2, \ldots, x_k))$$

  donde D es el dominio de definición de las variables $x_1, x_2, \ldots, x_k$ y P es una propiedad que cumplen las variables $x_1, x_2, \ldots, x_k$.

- $\exists$: "cuantificador existencial", con formato

$$\exists x_1, x_2, \ldots, x_k (D(x_1, x_2, \ldots, x_k) \wedge P(x_1, x_2, \ldots, x_k))$$

  donde D es el dominio de definición de las variables $x_1, x_2, \ldots, x_k$ y P es una propiedad que cumplen las variables $x_1, x_2, \ldots, x_k$.

A continuación, se van a considerar distintas maneras de definir formalmente los lenguajes que se han dado como ejemplos en el apartado 3.3.1.

El lenguaje L_1 es finito, contiene solo cuatro palabras. Además, no hay ninguna propiedad que caracterice a esas palabras, es decir, no hay ninguna propiedad que las distinga del resto de palabras de $\mathbb{A}^*$. Ello supone que no es posible definir L_1 mediante una propiedad, hay que dar todos sus componentes.

El lenguaje L_2 es infinito y está formado por todas las palabras definidas sobre el alfabeto $\mathbb{A} = \{a, b\}$ que no contienen ninguna aparición del símbolo b. Un lenguaje no puede ser definido formalmente utilizando puntos suspensivos. La manera formal de definir L_2 sería la siguiente:

$$L_2 = \{w \mid w \in \mathbb{A}^* \wedge \forall k((k \in \mathbb{N} \wedge 1 \leq k \leq |w|) \rightarrow w(k) = a)\}$$

Por tanto, L_2 puede ser definido como el conjunto de las palabras w de $\mathbb{A}^*$ que cumplen que para todo número natural comprendido entre 1 y la longitud de w, el símbolo de la posición correspondiente es a. Para la palabra vacía ε se cumple esa condición porque $|\varepsilon|$ es 0 y, por tanto, se tiene una fórmula universal con intervalo vacío. L_2 puede ser definido también de la siguiente forma:

$$L_2 = \{w \mid w \in \mathbb{A}^* \wedge \underbrace{|w|_b = 0}_{\substack{\text{ninguna} \\ \text{aparición} \\ \text{de } b}} \}$$

Una tercera opción para definir L_2 es la siguiente:

$$L_2 = \{w \mid w \in \mathbb{A}^* \wedge \underbrace{|w| = |w|_a}_{\substack{\text{longitud =} \\ \text{número de} \\ \text{apariciones} \\ \text{de } a}} \}$$

Una cuarta opción para definir L_2 es la siguiente:

$$L_2 = \{w \mid w \in \mathbb{A}^* \wedge \exists k(k \geq 0 \wedge w = (a\varepsilon)^k)\}$$

Una quinta opción para definir L_2 es la siguiente:

$$L_2 = \{w \mid w \in \mathbb{A}^* \wedge \neg es\_subpalabra(b\varepsilon, w)\}$$

Una sexta forma de definir L_2 es la siguiente:

$$L_2 = \{w \mid w \in \mathbb{A}^* \wedge \neg\exists v, u(v \in \mathbb{A}^* \wedge u \in \mathbb{A}^* \wedge w = v \cdot b\varepsilon \cdot u)\}$$

Para entender esta sexta definición de L_2, hay que tener en cuenta que si una palabra w contiene alguna aparición del símbolo b, esa palabra se puede dividir en tres subpalabras siendo la subpalabra del centro la palabra formada por el símbolo b, es decir, la palabra $b\varepsilon$. Por ejemplo, la palabra $aababaab\varepsilon$ se puede dividir de la siguiente forma: $\underbrace{aa\varepsilon}_{v} \cdot b\varepsilon \cdot \underbrace{abaab\varepsilon}_{u}$. Por tanto, esa palabra no pertenece a L_2. Esa misma palabra se puede dividir también de la siguiente manera: $\underbrace{aaba\varepsilon}_{v} \cdot b\varepsilon \cdot \underbrace{aab\varepsilon}_{u}$. Y también de esta otra forma: $\underbrace{aababaa\varepsilon}_{v} \cdot b\varepsilon \cdot \underbrace{\varepsilon}_{u}$. Pero si consideramos la palabra $aaaa\varepsilon$, esta palabra no se puede dividir en tres subpalabras de tal forma que la subpalabra del centro sea $b\varepsilon$ y como consecuencia de ello, $aaaa\varepsilon$ pertenece a L_2. Por tanto, L_2 está formado por las palabras que no admiten una división de la forma $v \cdot b\varepsilon \cdot u$.

El lenguaje L_3 es vacío, no contiene ninguna palabra. En vez de definir L_3 como $L_3 = \varnothing$, es posible definirlo también mediante una propiedad:

$$L_3 = \{w \mid w \in \mathbb{A}^* \wedge |w| = 0 \ \wedge \ |w| \neq 0\}$$

La propiedad $|w| = 0 \ \wedge \ |w| \neq 0$ es una propiedad que ninguna palabra puede cumplir y de esa forma indicamos que no hay ninguna palabra en L_3.

Otra manera de definir L_3 es la siguiente:

$$L_3 = \{w \mid w \in \mathbb{A}^* \wedge |w| \neq |w|\}$$

puesto que $|w|$ no puede ser distinto de $|w|$ y, consecuentemente, ninguna palabra w cumple $|w| \neq |w|$.

Otra manera adicional de definir L_3 es la siguiente:

$$L_3 = \{w \mid w \in \mathbb{A}^* \wedge |w| \leq \text{-1}\}$$

puesto que el conjunto de las palabras que tienen longitud negativa es vacío.

Otra manera más de definir L_3 es la siguiente:

$$L_3 = \{w \mid w \in \mathbb{A}^* \wedge \neg es\_subpalabra(\varepsilon, w)\}$$

puesto que ε es subpalabra de cualquier palabra y, por tanto, el conjunto de las palabras que no tienen a ε como subpalabra es vacío.

El lenguaje L_4 contiene un único elemento, la palabra vacía. Conviene tener muy claro que L_3 y L_4 son dos lenguajes distintos. Como L_4 es finito, una opción es dar los elementos que lo componen y la otra opción es definirlo mediante una propiedad que caracteriza a sus elementos. Una propiedad que caracteriza a ε es que su longitud es 0. Por tanto, podemos definir L_4 de la siguiente forma:

$$L_4 = \{w \mid w \in \mathbb{A}^* \wedge |w| = 0\}$$

El lenguaje L_5 está formado por las palabras de longitud 1. Al ser finito, se puede definir L_5 dando sus dos elementos o también mediante una propiedad. En este caso la propiedad que diferencia a las palabras de L_5 con respecto al resto de las palabras de $\mathbb{A}^*$ es que las palabras de L_5 tienen longitud 1:

$$L_5 = \{w \mid w \in \mathbb{A}^* \wedge |w| = 1\}$$

3.3.4 Función característica de un lenguaje

Un lenguaje L definido sobre un alfabeto $\mathbb{A}$, es un conjunto de elementos pertenecientes al universo $\mathbb{A}^*$. Consecuentemente, el lenguaje L puede ser descrito mediante su función característica [1] (o función indicatriz), que sería la siguiente:

$$\chi_L : \mathbb{A}^* \to \mathbb{2}$$

$$\chi_L(w) = \begin{cases} 1 & \text{si } w \in L \\ 0 & \text{si } w \notin L \end{cases}$$

donde $\mathbb{2}$ es el conjunto $\{0, 1\}$, 0 significa falso y 1 significa cierto. Por tanto, el conjunto o tipo $\mathbb{2}$ representa el tipo de los valores Booleanos.

La **función característica** de un lenguaje, es otra manera de dar la **especificación** de la tarea representada por ese lenguaje.

El hecho de que podamos formalizar la función característica de un lenguaje, no quiere decir que se pueda diseñar un algoritmo que sea capaz de representar el proceso a seguir para implementar dicha función característica. Tal como se verá más adelante, **hay funciones características que no pueden ser implementadas**. Es decir, en algunos casos, no es posible

[1] La función característica de un conjunto se representa, habitualmente, con la letra griega "ji" —equivalente a la jota del castellano— que se escribe χ. Ello es debido a que la palabra castellana *característica* viene de la palabra griega $\chi\alpha\rho\alpha\kappa\tau\eta\rho$, es decir, "jarakter".

diseñar un programa capaz de realizar el cálculo o la tarea especificada mediante la función característica. Por tanto, veremos, más adelante, que hay tareas que no pueden ser automatizadas o informatizadas.

Todo esto tiene consecuencias para la infomática, en general, y para el área de la informática que se ocupa del desarrollo de programas, en particular: Hay cálculos o tareas para los cuales no se puede desarrollar un programa.

Puesto que para que una palabra w de $\mathbb{A}^*$ pertenezca a L, deberá cumplir alguna propiedad concreta P que caracteriza a L, la definición de L en forma de conjunto tendrá, generalmente, la forma $L = \{w \mid w \in \mathbb{A}^* \wedge P(w)\}$, y la función característica tendrá la siguiente forma:

$$\chi_L : \mathbb{A}^* \to \mathbb{2}$$

$$\chi_L(w) = \begin{cases} 1 & \text{si } P(w) \\ \\ 0 & \text{en caso contrario} \end{cases}$$

Supongamos que tenemos el alfabeto $\mathbb{A} = \{a, b, c\}$ y los lenguajes L_1, L_2, L_3 y L_4 que se definen a continuación:

$$\begin{array}{l} L_1 = \{w \mid w \in \mathbb{A}^* \wedge |w|_b = 0 \wedge |w|_c = 0\} \\ L_2 = \{w \mid w \in \mathbb{A}^* \wedge |w| \leq 2\} \\ L_3 = \{w \mid w \in \mathbb{A}^* \wedge |w|_a = 8\} \\ L_4 = \{w \mid w \in \mathbb{A}^* \wedge (w = aa\varepsilon \vee w = acc\varepsilon \vee w = b\varepsilon)\} \end{array}$$

Esas definiciones constituyen las especificaciones de cuatro tareas.

L_1 es el lenguaje de las palabras que no contienen ni b ni c. Esa descripción de L_1 es una manera de dar la especificación de la tarea que consiste en detectar si una palabra del universo $\mathbb{A}^*$, cumple dicha propiedad. Otra forma de especificar esa tarea es mediante la función característica correspondiente:

$$\chi_{L_1} : \mathbb{A}^* \to \mathbb{2}$$

$$\chi_{L_1}(w) = \begin{cases} 1 & \text{si } |w|_b = 0 \wedge |w|_c = 0 \\ \\ 0 & \text{si } |w|_b \neq 0 \vee |w|_c \neq 0 \end{cases}$$

L_1 es un lenguaje infinito:

$$L_1 = \{\varepsilon, a\varepsilon, aa\varepsilon, aaa\varepsilon, \dots\}$$

Por su parte, L_2 es el lenguaje de las palabras cuya longitud es menor o igual que 2. Esa descripción de L_2 es una manera de dar la especificación de la tarea que consiste en detectar si una palabra del universo $\mathbb{A}^*$, cumple la mencionada propiedad. Esa tarea puede ser especificada también mediante la función característica correspondiente:

$$\chi_{L_2} : \mathbb{A}^* \to \mathbb{2}$$

$$\chi_{L_2}(w) = \begin{cases} 1 & \text{si } |w| \leq 2 \\ \\ 0 & \text{si } |w| \geq 3 \end{cases}$$

L_2 es un lenguaje finito:

$$L_2 = \{\varepsilon, a\varepsilon, b\varepsilon, c\varepsilon, aa\varepsilon, ab\varepsilon, ac\varepsilon, ba\varepsilon, bb\varepsilon, bc\varepsilon, ca\varepsilon, cb\varepsilon, cc\varepsilon\}$$

En cuanto a L_3, este es el lenguaje de las palabras que contienen exactamente 8 apariciones del símbolo a. No hay restricciones para los símbolos b y c. Esa descripción de L_3 es una manera de dar la especificación de la tarea que consiste en detectar si una palabra del universo $\mathbb{A}^*$, cumple la propiedad de contener, exactamente, 8 apariciones del símbolo a. Otra alternativa para dar la especificación de esa tarea es la función característica correspondiente:

$$\chi_{L_3} : \mathbb{A}^* \to \mathbb{2}$$

$$\chi_{L_3}(w) = \begin{cases} 1 & \text{si } |w|_a = 8 \\ \\ 0 & \text{si } |w|_a \neq 8 \end{cases}$$

L_3 es un lenguaje infinito:

$$\begin{aligned} L_3 = \ & \{aaaaaaaa\varepsilon, aaaaaaaab\varepsilon, aaaaaaaac\varepsilon, aaaaaaaba\varepsilon, aaaaaaaca\varepsilon, \ldots, \\ & baaaaaaaa\varepsilon, caaaaaaaa\varepsilon, aaaaaaaabb\varepsilon, aaaaaaaabc\varepsilon, aaaaaaaacb\varepsilon, \\ & aaaaaaaacc\varepsilon, \ldots\} \end{aligned}$$

Finalmente, L_4 es el lenguaje formado por las palabras $aa\varepsilon$, $acc\varepsilon$ y $b\varepsilon$. Esas palabras no cumplen ninguna propiedad general que las caracterice con respecto al resto de palabras del universo $\mathbb{A}^*$. De ahí que la propiedad que se ha dado expresa que se ha de ser alguna de las tres palabras en cuestión. La descripción que se ha dado de L_4, es una manera de dar la especificación de la tarea que consiste en detectar si una palabra del universo $\mathbb{A}^*$, es justo una de las tres palabras indicadas. Esa tarea puede ser especificada también mediante la función característica correspondiente:

$$\chi_{L_4} : \mathbb{A}^* \to \mathbb{2}$$

$$\chi_{L_4}(w) = \begin{cases} 1 & \text{si } (w = aa\varepsilon \vee w = acc\varepsilon \vee w = b\varepsilon) \\ \\ 0 & \text{en caso contrario} \end{cases}$$

L_4 es un lenguaje finito:

$$L_4 = \{a\varepsilon, acc\varepsilon, b\varepsilon\}$$

3.3.5 Definición del conjunto de todos los lenguajes definibles sobre el alfabeto $\mathbb{A}$: los conjuntos $\mathsf{Sub}(\mathbb{A}^*)$ y $\mathbb{2}^{\mathbb{A}^*}$

Los lenguajes definibles sobre un alfabeto $\mathbb{A}$ se pueden describir de dos maneras: como conjunto o mediante la función característica correspondiente.

Si optamos por representar los lenguajes como conjuntos, el conjunto de todos los lenguajes definibles sobre el alfabeto $\mathbb{A}$ lo denotaremos como $\mathsf{Sub}(\mathbb{A}^*)$, es decir, como el conjunto formado por todos los subconjuntos de $\mathbb{A}^*$. Nótese que $\mathbb{A}^*$ es también un subconjunto de $\mathbb{A}^*$. [2]

Si $\mathbb{A} = \{a, b\}$, entonces $\mathsf{Sub}(\mathbb{A}^*)$, informalmente, viene a ser el siguiente conjunto de conjuntos:

$$\{\varnothing, \{\varepsilon\}, \{a\varepsilon\}, \{b\varepsilon\}, \{c\varepsilon\}, \{\varepsilon, a\varepsilon, aa\varepsilon, b\varepsilon\}, \{a\varepsilon, b\varepsilon, cba\varepsilon\}, \{a\varepsilon, aa\varepsilon, aaa\varepsilon, \dots\}, \dots\}$$

donde están todos los conjuntos que se pueden formar con elementos de $\mathbb{A}^*$, es decir, $\mathsf{Sub}(\mathbb{A}^*)$ es el conjunto de todos los lenguajes definibles sobre $\mathbb{A}$.

El conjunto $\mathsf{Sub}(\mathbb{A}^*)$ es siempre infinito.

Si optamos por representar cada lenguaje L mediante su función característica $\chi_L : \mathbb{A}^* \to \mathbb{2}$, el conjunto de todos los lenguajes definibles sobre el alfabeto $\mathbb{A}$, puede ser expresado como el conjunto de las funciones características correspondientes a esos lenguajes. Dicho conjunto se denota como $\mathbb{2}^{\mathbb{A}^*}$ y, tal como se acaba de indicar, representa el conjunto de todas las funciones de la forma $\mathbb{A}^* \to \mathbb{2}$.

Por tanto, $\mathsf{Sub}(\mathbb{A}^*)$ y $\mathbb{2}^{\mathbb{A}^*}$ representan lo mismo. De aquí en adelante, se utilizará la notación $\mathbb{2}^{\mathbb{A}^*}$ y no se utilizará la notación $\mathsf{Sub}(\mathbb{A}^*)$.

El conjunto $\mathbb{2}^{\mathbb{A}^*}$ es siempre infinito.

3.3.6 Operaciones sobre lenguajes

Teniendo en cuenta la definición, los lenguajes son conjuntos de secuencias de símbolos. Por tanto, algunas de las operaciones que se definen sobre lenguajes, son adaptaciones de las operaciones habituales sobre conjuntos y, otras operaciones, están basadas en operaciones sobre palabras.

Conviene recordar que, matemáticamente, un conjunto no contiene elementos repetidos, es decir, cada elemento aparece solo una vez. Por ejemplo, si C es un conjunto de palabras

[2] Un conjunto C es subconjunto de otro conjunto D si todos los elementos de C pertenecen a D. Puesto que todos los elementos de D pertenecen a D, tenemos que D es subconjunto de D.

(es decir, un lenguaje) y la palabra $abba\varepsilon$ pertenece a C, la palabra $abba\varepsilon$ solo aparece una vez en C. Por tanto, al representar un conjunto, lo habitual es no poner elementos repetidos. De todas formas, es posible repetir elementos al representar un conjunto, pero si permitimos repeticiones, nos encontramos con que $\{aa\varepsilon, bab\varepsilon, aa\varepsilon, aa\varepsilon, b\varepsilon, aa\varepsilon, b\varepsilon\}$ y $\{bab\varepsilon, aa\varepsilon, b\varepsilon\}$ representan el mismo conjunto y esas dos representaciones han de ser consideradas equivalentes. Además, matemáticamente, no hay orden en los conjuntos y, por consiguiente, la expresión $\{aa\varepsilon, bab\varepsilon, b\varepsilon\}$ y la expresión $\{bab\varepsilon, aa\varepsilon, b\varepsilon\}$ representan el mismo conjunto.

A continuación se presentan diez operaciones sobre lenguajes:

1. Unión: $L_1 \cup L_2$
2. Intersección: $L_1 \cap L_2$
3. Diferencia: $L_1 \setminus L_2$
4. Complementario: $\overline{L}$
5. Concatenación: $L_1 \circ L_2$
6. Exponenciación: L^k
7. Clausura universal: L^*
8. Clausura positiva: L^+
9. Inverso: L^R
10. Cociente: $L_1 \setminus\!\setminus L_2$

3.3.6.1 Unión

Sean L_1 y L_2 dos lenguajes definidos sobre el alfabeto $\mathbb{A}$. El lenguaje que se obtiene al unir L_1 y L_2 es el lenguaje formado por las palabras que pertenecen a L_1 o a L_2, por lo menos a uno de ellos.

$$L_1 \cup L_2 = \{w \mid w \in \mathbb{A}^* \wedge (w \in L_1 \vee w \in L_2)\}$$

El tipo de la unión es el siguiente:

$$2^{\mathbb{A}^*} \times 2^{\mathbb{A}^*} \to 2^{\mathbb{A}^*}$$

El tipo indica que, dados dos subconjuntos de $\mathbb{A}^*$, se obtiene otro subconjunto de $\mathbb{A}^*$. Dicho de otra forma, dados dos elementos de $2^{\mathbb{A}^*}$, se obtiene otro elemento de $2^{\mathbb{A}^*}$. Es decir, dados dos lenguajes, se obtiene otro lenguaje.

Puesto que un lenguaje puede ser expresado también mediante su función característica, las operaciones sobre lenguajes pueden ser definidas en términos de las funciones características

correspondientes.

Si χ_{L_1} y χ_{L_2} son las funciones características de los lenguajes L_1 y L_2, respectivamente, entonces la función característica de $L_1 \cup L_2$ puede ser expresada de la siguiente forma:

$$\chi_{L_1 \cup L_2} : \mathbb{A}^* \to \mathbb{2}$$

$$\chi_{L_1 \cup L_2}(w) = \begin{cases} 1 & \text{si } \chi_{L_1}(w) = 1 \ \vee \ \chi_{L_2}(w) = 1 \\ 0 & \text{si } \chi_{L_1}(w) = 0 \ \wedge \ \chi_{L_2}(w) = 0 \end{cases}$$

Propiedades de la unión:

- La unión es **asociativa**. Para tres lenguajes cualesquiera L_1, L_2 y L_3 se cumple

$$L_1 \cup (L_2 \cup L_3) = (L_1 \cup L_2) \cup L_3$$

  Por tanto, los paréntesis no son necesarios: $L_1 \cup L_2 \cup L_3$

- La unión es **conmutativa**. Para dos lenguajes cualesquiera L_1 y L_2 se cumple

$$L_1 \cup L_2 = L_2 \cup L_1$$

- El lenguaje vacío es el **elemento neutro** para $\cup$. Por tanto, para cualquier lenguaje L se cumple

$$\varnothing \cup L = L \cup \varnothing = L$$

- Para cualquier lenguaje L se cumple $L \cup L = L$ y $L \cup \mathbb{A}^* = \mathbb{A}^*$.

Recordemos que $\varnothing \neq \{\varepsilon\}$. Por tanto, dado un lenguaje cualquiera L, en general $L \cup \{\varepsilon\} \neq L$. Para que se cumpla $L \cup \{\varepsilon\} = L$, la palabra vacía ε tiene que pertenecer a L.

3.3.6.2 Intersección

Sean L_1 y L_2 dos lenguajes definidos sobre el alfabeto $\mathbb{A}$. La intersección entre L_1 y L_2 es el lenguaje formado por las palabras de L_1 que también están en L_2.

$$L_1 \cap L_2 = \{w \mid w \in \mathbb{A}^* \wedge w \in L_1 \wedge w \in L_2\}$$

El tipo de la intersección es el siguiente:

$$\mathbb{2}^{\mathbb{A}^*} \times \mathbb{2}^{\mathbb{A}^*} \to \mathbb{2}^{\mathbb{A}^*}$$

El tipo indica que dados dos subconjuntos de $\mathbb{A}^*$, se obtiene otro subconjunto de $\mathbb{A}^*$. Dicho de otra forma, dados dos elementos de $2^{\mathbb{A}^*}$, se obtiene otro elemento de $2^{\mathbb{A}^*}$. Es decir, dados dos lenguajes, se obtiene otro lenguaje.

Si χ_{L_1} y χ_{L_2} son las funciones características de los lenguajes L_1 y L_2, respectivamente, entonces la función característica de $L_1 \cap L_2$ puede ser expresada de la siguiente forma:

$$\chi_{L_1 \cap L_2} : \mathbb{A}^* \to 2$$

$$\chi_{L_1 \cap L_2}(w) = \begin{cases} 1 & \text{si } \chi_{L_1}(w) = 1 \ \wedge \ \chi_{L_2}(w) = 1 \\ 0 & \text{si } \chi_{L_1}(w) = 0 \ \vee \ \chi_{L_2}(w) = 0 \end{cases}$$

Propiedades de la intersección:

- La intersección es **asociativa**. Para tres lenguajes cualesquiera L_1, L_2 y L_3 se cumple

$$L_1 \cap (L_2 \cap L_3) = (L_1 \cap L_2) \cap L_3$$

  Por tanto, los paréntesis no son necesarios: $L_1 \cap L_2 \cap L_3$

- La intersección es **conmutativa**. Para dos lenguajes cualesquiera L_1 y L_2 se cumple

$$L_1 \cap L_2 = L_2 \cap L_1$$

- El lenguaje universal $\mathbb{A}^*$ es el **elemento neutro** para $\cap$. Por tanto, para cualquier lenguaje L se cumple

$$\mathbb{A}^* \cap L = L \cap \mathbb{A}^* = L$$

- Para cualquier lenguaje L se cumple $L \cap \varnothing = \varnothing$ y $L \cap L = L$.

En lo que respecta al lenguaje $\{\varepsilon\}$, dado un lenguaje cualquiera L, se cumple $L \cap \{\varepsilon\} = \{\varepsilon\}$ si $\varepsilon \in L$. Por el contrario, $L \cap \{\varepsilon\} = \varnothing$ si $\varepsilon \notin L$.

3.3.6.3 Diferencia

Sean L_1 y L_2 dos lenguajes definidos sobre el alfabeto $\mathbb{A}$. La diferencia entre L_1 y L_2 es el lenguaje formado por las palabras de L_1 que no pertenecen a L_2.

$$L_1 \setminus L_2 = \{w \mid w \in \mathbb{A}^* \wedge w \in L_1 \wedge w \notin L_2\}$$

El tipo de la diferencia es el siguiente:

$$2^{\mathbb{A}^*} \times 2^{\mathbb{A}^*} \to 2^{\mathbb{A}^*}$$

El tipo indica que dados dos subconjuntos de $\mathbb{A}^*$, se obtiene otro subconjunto de $\mathbb{A}^*$. Dicho de otra forma, dados dos elementos de $2^{\mathbb{A}^*}$, se obtiene otro elemento de $2^{\mathbb{A}^*}$. Es decir, dados dos lenguajes, se obtiene otro lenguaje.

Si χ_{L_1} y χ_{L_2} son las funciones características de los lenguajes L_1 y L_2, respectivamente, entonces la función característica de $L_1 \setminus L_2$ puede ser expresada de la siguiente forma:

$$\chi_{L_1 \setminus L_2} : \mathbb{A}^* \to 2$$

$$\chi_{L_1 \setminus L_2}(w) = \begin{cases} 1 & \text{si } \chi_{L_1}(w) = 1 \ \wedge \ \chi_{L_2}(w) = 0 \\ 0 & \text{si } \chi_{L_1}(w) = 0 \ \vee \ \chi_{L_2}(w) = 1 \end{cases}$$

Propiedades de la diferencia:

- La diferencia **no es asociativa**. Veamos un contraejemplo. Sean $L_1 = \{aa\varepsilon, ab\varepsilon, ba\varepsilon, bb\varepsilon\}$, $L_2 = \{aa\varepsilon, aaa\varepsilon\}$ y $L_3 = \{aa\varepsilon, bab\varepsilon, a\varepsilon\}$:

  $$L_1 \setminus (L_2 \setminus L_3) \neq (L_1 \setminus L_2) \setminus L_3$$

  Ya que $L_1 \setminus (L_2 \setminus L_3) = L_1$ mientras que $(L_1 \setminus L_2) \setminus L_3 = \{ab\varepsilon, ba\varepsilon, bb\varepsilon\}$

- La diferencia **no es conmutativa**. Como contraejemplo consideremos $L_1 = \{aa\varepsilon, ab\varepsilon, ba\varepsilon, bb\varepsilon\}$ y $L_2 = \{aa\varepsilon, aaa\varepsilon\}$:

  $$L_1 \setminus L_2 \neq L_2 \setminus L_1$$

  Ya que $L_1 \setminus L_2 = \{ab\varepsilon, ba\varepsilon, bb\varepsilon\}$ y $L_2 \setminus L_1 = \{aaa\varepsilon\}$.

- Para cualquier lenguaje L se cumplen $L \setminus \varnothing = L$, $L \setminus L = \varnothing$ y $L \setminus \mathbb{A}^* = \varnothing$.

3.3.6.4 Complementario

Sea L un lenguaje definido sobre el alfabeto $\mathbb{A}$. El complementario de L es denotado como $\overline{L}$ y es el lenguaje formado por las palabras de $\mathbb{A}^*$ que no pertenecen a L.

$$\overline{L} = \{w \mid w \in \mathbb{A}^* \wedge w \notin L\}$$

Otra manera de definir el complementario es utilizando la diferencia: $\overline{L} = \mathbb{A}^* \setminus L$.

Por ejemplo, consideremos el lenguaje $L = \{w \mid w \in \mathbb{A}^* \wedge |w| \geq 3\}$ definido sobre el alfabeto $\mathbb{A} = \{a, b\}$. El lenguaje infinito L contiene todas las palabras definibles sobre el alfabeto $\mathbb{A}$ que tengan longitud mayor o igual que 3. El complementario de L sería $\overline{L} = \{\varepsilon, a\varepsilon, b\varepsilon, aa\varepsilon, ab\varepsilon, ba\varepsilon, bb\varepsilon\}$, es decir, el lenguaje formado por las palabras de $\mathbb{A}^*$ que no están en L. Por tanto, $\overline{L}$ es finito en este ejemplo, pero puede ocurrir que el complementario de un lenguaje infinito sea también infinito. Por ejemplo, consideremos es lenguaje infinito

$H = \{w \mid w \in \mathbb{A}^* \wedge |w| = |w|_a\}$ definido sobre el alfabeto $\mathbb{A} = \{a, b\}$. El lenguaje H está formado por todas las palabras definidas sobre $\mathbb{A}$ que no contienen ninguna b. Su complementario, el lenguaje $\overline{H}$, es un lenguaje infinito.

El tipo de la función que calcula el complementario es el siguiente:

$$2^{\mathbb{A}^*} \to 2^{\mathbb{A}^*}$$

Es decir, dado un subconjunto de $\mathbb{A}^*$, se obtiene otro subconjunto de $\mathbb{A}^*$. En otras palabras, dado un elemento de $2^{\mathbb{A}^*}$, se obtiene otro elemento de $2^{\mathbb{A}^*}$. Es decir, dado un lenguaje, se obtiene otro lenguaje.

Si χ_L es la función característica del lenguaje L, entonces la función característica de $\overline{L}$ puede ser expresada de la siguiente forma:

$$\chi_{\overline{L}} : \mathbb{A}^* \to 2$$

$$\chi_{\overline{L}}(w) = \begin{cases} 1 & \text{si } \chi_L(w) = 0 \\ 0 & \text{si } \chi_L(w) = 1 \end{cases}$$

Propiedades del complemento:

- Para cualquier lenguaje L se cumple $\overline{\overline{L}} = L$.
- Para dos lenguajes cualesquiera L_1 y L_2 se cumple $\overline{L_1 \cup L_2} = \overline{L_1} \cap \overline{L_2}$.
- Para dos lenguajes cualesquiera L_1 y L_2 se cumple $\overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$.
- $\overline{\varnothing} = \mathbb{A}^*$ y $\overline{\mathbb{A}^*} = \varnothing$.

3.3.6.5 Concatenación

Sean L_1 y L_2 dos lenguajes definidos sobre el alfabeto $\mathbb{A}$. La concatenación entre L_1 y L_2 es el lenguaje denotado como $L_1 \circ L_2$ y formado por todas las palabras que se obtienen al concatenar o juntar una palabra de L_1 y otra de L_2 (en ese orden). El operador de la operación de concatenación de lenguajes es $\circ$.

$$L_1 \circ L_2 = \{w \mid w \in \mathbb{A}^* \wedge \exists u, v(u \in L_1 \wedge v \in L_2 \wedge w = u \cdot v)\}$$

Por ejemplo, si consideramos los lenguajes $L_1 = \{aa\varepsilon, cc\varepsilon\}$ y $L_2 = \{bcb\varepsilon, c\varepsilon\}$, el lenguaje $L_1 \circ L_2$ es $\{aa\varepsilon \cdot bcb\varepsilon, aa\varepsilon \cdot c\varepsilon, cc\varepsilon \cdot bcb\varepsilon, cc\varepsilon \cdot c\varepsilon\}$. Por tanto, $L_1 \circ L_2 = \{\underline{aa}bcb\varepsilon, \underline{aa}c\varepsilon, \underline{cc}bcb\varepsilon, \underline{cc}c\varepsilon\}$, donde los trozos correspondientes a las palabras del lenguaje L_1 aparecen subrayados para mejorar la legibilidad.

El tipo de la concatenación es el siguiente:

$$2^{\mathbb{A}^*} \times 2^{\mathbb{A}^*} \to 2^{\mathbb{A}^*}$$

El tipo indica que, dados dos subconjuntos de $\mathbb{A}^*$, se obtiene otro subconjunto de $\mathbb{A}^*$. Dicho de otra forma, dados dos elementos de $2^{\mathbb{A}^*}$, se obtiene otro elemento de $2^{\mathbb{A}^*}$. Es decir, dados dos lenguajes, se obtiene otro lenguaje.

Si χ_{L_1} y χ_{L_2} son las funciones características de los lenguajes L_1 y L_2, respectivamente, entonces la función característica de $L_1 \circ L_2$ puede ser expresada de la siguiente forma:

$$\chi_{L_1 \circ L_2} : \mathbb{A}^* \to 2$$

$$\chi_{L_1 \circ L_2}(w) = \begin{cases} 1 & \text{si } \exists u, v(u \in \mathbb{A}^* \wedge v \in \mathbb{A}^* \wedge \chi_{L_1}(u) = 1 \ \wedge \ \chi_{L_2}(v) = 1 \wedge w = u \cdot v) \\ 0 & \text{en caso contrario} \end{cases}$$

Propiedades de la concatenación:

- La concatenación es **asociativa**. Para tres lenguajes cualesquiera L_1, L_2 y L_3 se cumple

  $$L_1 \circ (L_2 \circ L_3) = (L_1 \circ L_2) \circ L_3$$

  Por tanto, los paréntesis no son necesarios: $L_1 \circ L_2 \circ L_3$

- La concatenación **no es conmutativa**. Como contraejemplo, tenemos que para los lenguajes $L_1 = \{aa\varepsilon, cc\varepsilon\}$ y $L_2 = \{bcb\varepsilon, b\varepsilon\}$ se cumple

  $$L_1 \circ L_2 = \{aa\varepsilon \cdot bcb\varepsilon, aa\varepsilon \cdot c\varepsilon, cc\varepsilon \cdot bcb\varepsilon, cc\varepsilon \cdot c\varepsilon\} = \{\underline{aa}bcb\varepsilon, \underline{aa}c\varepsilon, \underline{cc}bcb\varepsilon, \underline{cc}c\varepsilon\}$$
  $$L_2 \circ L_1 = \{bcb\varepsilon \cdot aa\varepsilon, bcb\varepsilon \cdot cc\varepsilon, b\varepsilon \cdot aa\varepsilon, b\varepsilon \cdot cc\varepsilon\} = \{bcb\underline{aa}\varepsilon, bcb\underline{cc}\varepsilon, b\underline{aa}\varepsilon, b\underline{cc}\varepsilon\}$$

  Los trozos correspondientes a las palabras del lenguaje L_1 aparecen subrayados para mejorar la legibilidad.

- El lenguaje de la palabra vacía $\{\varepsilon\}$ es el **elemento neutro** para la concatenación. Por tanto, para cualquier lenguaje L se cumple

  $$\{\varepsilon\} \circ L = L \circ \{\varepsilon\} = L$$

- Para cualquier lenguaje L se cumple $L \circ \varnothing = \varnothing \circ L = \varnothing$.

- $\mathbb{A}^* \circ \mathbb{A}^* = \mathbb{A}^*$.

3.3.6.6 Exponenciación

Sea L un lenguaje definido sobre el alfabeto $\mathbb{A}$ y k un número natural. El lenguaje L^k se define como:

- $\{\varepsilon\}$ si $k = 0$
- $L \circ L^{k-1}$ si $k \geq 1$.

En definitiva, L^k se obtiene concatenando L k veces: $\underbrace{L \circ L \circ \cdots \circ L}_{k \text{ veces}}$. Por tanto, una palabra w que pertenece a L^k se puede descomponer en k subpalabras $w_1, w_2, \ldots, w_k$ tal que $w = w_1 \cdot w_2 \cdot \ \ldots \ \cdot w_k$ y $w_j \in L$ para todo $j \in \{1, \ldots, k\}$, es decir, que todas las palabras $w_1, w_2, \ldots, w_k$ pertenecen a L.

Cuando k es mayor o igual que 1, L^k se puede definir también de esta forma:

$$L^k = \{w \mid w \in \mathbb{A}^* \wedge \exists u, v(u \in L \wedge v \in L^{k-1} \wedge w = u \cdot v)\}$$

Por tanto, si $k \geq 1$, entonces L^k contiene las palabras w que se pueden descomponer en dos subpalabras u y v tal que u pertenece a L y v pertenece a L^{k-1}.

Por ejemplo, si consideramos el lenguaje $L = \{ab\varepsilon, cb\varepsilon\}$, el lenguaje L^3 es el lenguaje $L \circ L \circ L$, es decir, $\{ababab\varepsilon, ababcb\varepsilon, abcbab\varepsilon, abcbcb\varepsilon, cbabab\varepsilon, cbabcb\varepsilon, cbcbab\varepsilon, cbcbcb\varepsilon\}$. El lenguaje L^3 contiene las 8 maneras de generar tripletas combinando las palabras $ab\varepsilon$ y $cb\varepsilon$.

El tipo de la exponenciación es el siguiente:

$$2^{\mathbb{A}^*} \times \mathbb{N} \to 2^{\mathbb{A}^*}$$

Ese tipo indica que, dados un subconjunto de $\mathbb{A}^*$ y un natural (un entero mayor o igual que 0), se obtiene otro subconjunto de $\mathbb{A}^*$. En otras palabras, dados un elemento de $2^{\mathbb{A}^*}$ y un número natural, se obtiene otro elemento de $2^{\mathbb{A}^*}$. Es decir, dados un lenguaje y un número natural, se obtiene otro lenguaje.

En cuanto a la notación basada en el concepto de función característica, se tiene una función caracterśtica por cada lenguaje $L \in 2^{\mathbb{A}^*}$ y cada número natural $k \in \mathbb{N}$.

En el caso de que k sea 0, la función característica para cualquier lenguaje L es la siguiente:

$$\chi_{L^0} : \mathbb{A}^* \to 2$$

$$\chi_{L^0}(w) = \begin{cases} 1 & \text{si } w = \varepsilon \\ 0 & \text{en caso contrario} \end{cases}$$

En el caso de que k sea ≥ 1, la función característica para un lenguaje L^k depende de la función característica de L y de la función característica de L^{k-1}:

$$\chi_{L^k} : \mathbb{A}^* \to \mathbb{2}$$

$$\chi_{L^k}(w) = \begin{cases} 1 & \text{si } \exists u, v(u \in \mathbb{A}^* \wedge v \in \mathbb{A}^* \wedge \chi_L(u) = 1 \wedge \chi_{L^{k-1}}(v) = 1 \wedge w = u \cdot v) \\ 0 & \text{en caso contrario} \end{cases}$$

Propiedades de la exponenciación:

- Para cualquier lenguaje L y para dos números naturales cualesquiera j y k se cumple: $L^k \circ L^j = L^{k+j}$.
- Para cualquier lenguaje L y para dos números naturales cualesquiera j y k se cumple: $(L^k)^j = L^{k*j}$, donde $*$ representa la multiplicación o producto entre números naturales.
- $\{\varepsilon\}^k = \{\varepsilon\}$ para cualquier número natural $k \geq 0$.
- $\varnothing^0 = \{\varepsilon\}$ y $\varnothing^k = \varnothing$ para cualquier número natural $k \geq 1$.
- $L^1 = L$ y $L^0 = \{\varepsilon\}$ para cualquier lenguaje L.
- $(\mathbb{A}^*)^0 = \{\varepsilon\}$.
- $(\mathbb{A}^*)^k = \mathbb{A}^*$ para cualquier número natural $k \geq 1$.

3.3.6.7 Clausura universal

Sea L un lenguaje definido sobre el alfabeto $\mathbb{A}$. La **clausura universal** de L se denota como L^* y se define como:

$$L^* = \bigcup_{k \in \mathbb{N}} L^k$$

De manera informal, la definición sería la siguiente:

$$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \ldots$$

Por tanto, L^* es la unión de infinitos conjuntos. En concreto, la unión de todas las potencias posibles de L.

Otra manera formal de definir L^* es la siguiente:

$$L^* = \{w \mid w \in \mathbb{A}^* \wedge \exists k(k \in \mathbb{N} \wedge w \in L^k)\}$$

L^* es un lenguaje definido sobre el alfabeto $\mathbb{A}$ y, por tanto, es un subconjunto de $\mathbb{A}^*$, es decir, $L^* \subseteq \mathbb{A}^*$.

Por ejemplo, si consideramos el lenguaje $L = \{aa\varepsilon\}$ definido sobre el alfabeto $\mathbb{A} = \{a, b\}$, el lenguaje L^* contiene las palabras que tienen un número par de a's y ninguna b:

$$L^* = \{\varepsilon, aa\varepsilon, aaaa\varepsilon, aaaaaa\varepsilon, \dots\} = \{w \mid w \in \mathbb{A}^* \wedge \underbrace{|w| = |w|_a}_{\text{solo hay } a\text{'s}} \wedge \underbrace{|w| \bmod 2 = 0}_{\text{número par de } a\text{'s}}\}$$

Por tanto, L^* no contiene, por ejemplo, la palabra $aba\varepsilon$ que sí está en $\mathbb{A}^*$. De esta forma, tenemos un ejemplo en el que se ve que $L^* \subsetneq \mathbb{A}^*$, es decir, el lenguaje L^* está estrictamente contenido en $\mathbb{A}^*$.

El tipo de la clausura universal es el siguiente:

$$2^{\mathbb{A}^*} \to 2^{\mathbb{A}^*}$$

Es decir, dado un subconjunto de $\mathbb{A}^*$, se obtiene otro subconjunto de $\mathbb{A}^*$. En otras palabras, dado un elemento de $2^{\mathbb{A}^*}$, se obtiene otro elemento de $2^{\mathbb{A}^*}$. Esto significa que, dado un lenguaje, se obtiene otro lenguaje.

En cuanto a la notación basada en el concepto de función característica, dado un lenguaje L, la función característica para el lenguaje L^* depende de las funciones características correspondientes a los lenguajes L^k, con $k \in \mathbb{N}$:

$$\chi_{L^*} : \mathbb{A}^* \to 2$$

$$\chi_{L^*}(w) = \begin{cases} 1 & \text{si } \exists k(k \in \mathbb{N} \ \wedge \ \chi_{L^k}(w) = 1) \\ 0 & \text{en caso contrario} \end{cases}$$

Propiedades de la clausura:

- $(L^*)^* = L^*$.
- $\{\varepsilon\}^* = \{\varepsilon\}$.
- $\varnothing^* = \{\varepsilon\}$.

Si consideramos el alfabeto $\mathbb{A} = \{a, b, c\}$, tenemos que $\mathbb{A}^* = \{a\varepsilon, b\varepsilon, c\varepsilon\}^*$ y $(\mathbb{A}^*)^* = (\{a\varepsilon, b\varepsilon, c\varepsilon\}^*)^* = \{a\varepsilon, b\varepsilon, c\varepsilon\}^* = \mathbb{A}^*$. Por tanto, $(\mathbb{A}^*)^* = \mathbb{A}^*$.

Para un alfabeto $\mathbb{A}$ cualquiera, tenemos lo siguiente:

$$\begin{aligned} \mathbb{A}^* &= \{\alpha\varepsilon \mid \alpha \in \mathbb{A}\}^* \\ (\mathbb{A}^*)^* &= (\{\alpha\varepsilon \mid \alpha \in \mathbb{A}\}^*)^* \\ &= \{\alpha\varepsilon \mid \alpha \in \mathbb{A}\}^* \text{ por la propiedad } (L^*)^* = L^* \text{ para cualquier } L \\ &= \mathbb{A}^* \end{aligned}$$

3.3.6.8 Clausura positiva

Sea L un lenguaje definido sobre el alfabeto $\mathbb{A}$. La **clausura positiva** de L se denota como L^+ y se define como:

$$L^+ = \bigcup_{k \in \mathbb{N} \wedge k \geq 1} L^k$$

De manera informal, la definición sería la siguiente:

$$L^+ = L^1 \cup L^2 \cup L^3 \ldots$$

Una alternativa formal para definir la clausura positiva de L es la siguiente:

$$L^+ = L \circ L^*$$

El tipo de la clausura positiva es el mismo que el de la clausura universal.

$$\mathbb{2}^{\mathbb{A}^*} \to \mathbb{2}^{\mathbb{A}^*}$$

El tipo indica que, dado un lenguaje, se obtiene otro lenguaje.

Dado un lenguaje L, la función característica para el lenguaje L^+ depende de las funciones características correspondientes a los lenguajes L^k, con $k \in \mathbb{N}$ y $k \geq 1$:

$$\chi_{L^+} : \mathbb{A}^* \to \mathbb{2}$$

$$\chi_{L^+}(w) = \begin{cases} 1 & \text{si } \exists k(k \in \mathbb{N} \ \wedge \ k \geq 1 \ \wedge \ \chi_{L^k}(w) = 1) \\ 0 & \text{en caso contrario} \end{cases}$$

Aunque al definir L^+ no se tiene en cuenta L^0, es decir, no se añade el conjunto $\{\varepsilon\}$, la palabra vacía ε puede pertenecer a L^+. Esto será así si ε pertenece a L. Por tanto, siempre se cumple $L^* = L^+ \cup \{\varepsilon\}$ pero en algunos casos $L^* \setminus \{\varepsilon\}$ no es igual a L^+, ya que si ε pertenece a L^+, entonces $L^* \setminus \{\varepsilon\} \neq L^+$.

Para cualquier lenguaje L, se cumple $L^+ \subseteq L^*$ y, por consiguiente, L^+ es un lenguaje definido sobre el alfabeto $\mathbb{A}$, es decir, L^+ es un subconjunto de $\mathbb{A}^*$.

Por ejemplo, si consideramos el lenguaje $L = \{aa\varepsilon\}$ definido sobre el alfabeto $\mathbb{A} = \{a, b\}$, el lenguaje L^+ contiene las palabras que tienen por lo menos una aparición de a, un número par de a's y ninguna aparición de b:

$$\begin{aligned} L^+ &= \{aa\varepsilon, aaaa\varepsilon, aaaaaa\varepsilon, \ldots\} \\ &= \{w \mid w \in \mathbb{A}^* \ \wedge \ \underbrace{|w|_a \geq 1}_{\substack{\text{al menos una} \\ \text{aparición} \\ \text{de } a}} \ \wedge \ \underbrace{|w| = |w|_a}_{\substack{\text{solo} \\ \text{hay } a\text{'s}}} \ \wedge \ \underbrace{|w| \bmod 2 = 0}_{\substack{\text{número par} \\ \text{de } a\text{'s}}}\} \end{aligned}$$

En este ejemplo, L^+ no contiene ε y por tanto $L^+ \neq L^*$.

Para el caso del lenguaje vacío, se cumple $\varnothing^+ = \varnothing$. Además, tenemos que $(\mathbb{A}^*)^+ = \mathbb{A}^*$.

3.3.6.9 Inverso

Sea L un lenguaje definido sobre el alfabeto $\mathbb{A}$. El inverso de L es denotado como L^R y es el lenguaje formado por las palabras de $\mathbb{A}^*$ que son inversas de alguna palabra de L.

$$L^R = \{w \mid w \in \mathbb{A}^* \ \wedge \ w^R \in L\} \ ^3$$

Por ejemplo, si consideramos el lenguaje $L = \{\varepsilon, a\varepsilon, ab\varepsilon, aab\varepsilon, ac\varepsilon\}$ definido sobre el alfabeto $\mathbb{A} = \{a, b, c\}$, tenemos que $L^R = \{\varepsilon, a\varepsilon, ba\varepsilon, baa\varepsilon, ca\varepsilon\}$.

El tipo de la función que calcula el inverso es el siguiente:

$$2^{\mathbb{A}^*} \rightarrow 2^{\mathbb{A}^*}$$

El tipo indica que, dado un subconjunto de $\mathbb{A}^*$, se obtiene otro subconjunto de $\mathbb{A}^*$. Dicho de otra forma, dado un elemento de $2^{\mathbb{A}^*}$, se obtiene otro elemento de $2^{\mathbb{A}^*}$. Es decir, dado un lenguaje, se obtiene otro lenguaje.

Dado un lenguaje L, la función característica para el lenguaje L^R depende de la función característica correspondiente al lenguaje L:

$$\chi_{L^R} : \mathbb{A}^* \rightarrow 2$$

$$\chi_{L^R}(w) = \begin{cases} 1 & \text{si } \chi_L(w^R) = 1 \\ 0 & \text{en caso contrario} \end{cases}$$

Propiedades del inverso:

- Para cualquier lenguaje L se cumple $(L^R)^R = L$.
- Para dos lenguajes cualesquiera L_1 y L_2 se cumple $(L_1 \cup L_2)^R = L_1^R \cup L_2^R$.
- Para dos lenguajes cualesquiera L_1 y L_2 se cumple $(L_1 \cap L_2)^R = L_1^R \cap L_2^R$.
- Para dos lenguajes cualesquiera L_1 y L_2 se cumple $(L_1 \circ L_2)^R = L_2^R \circ L_1^R$.
  Nótese que cambia el orden.
- $\varnothing^R = \varnothing$ y $\{\varepsilon\}^R = \{\varepsilon\}$.
- $(\mathbb{A}^*)^R = \mathbb{A}^*$.

[3] R proviene de la palabra inglesa “Reverse”

3.3.6.10 Cociente

Sean L_1 y L_2 dos lenguajes definidos sobre el alfabeto $\mathbb{A}$. El *cociente* de L_1 y L_2 es denotado como $L_1 \setminus\!\setminus L_2$ y es el lenguaje formado por las palabras de $\mathbb{A}^*$ que al ser concatenadas con alguna palabra de L_2 dan lugar a palabras de L_1.

$$L_1 \setminus\!\setminus L_2 = \{w \mid w \in \mathbb{A}^* \wedge \exists u, v(u \in L_1 \wedge v \in L_2 \wedge u = w \cdot v)\}$$

Por ejemplo, si consideramos el lenguaje $L_1 = \{a\varepsilon, ab\varepsilon, aab\varepsilon, aac\varepsilon, cc\varepsilon\}$ y el lenguaje $L_2 = \{a\varepsilon, bbb\varepsilon, c\varepsilon\}$ definidos sobre el alfabeto $\mathbb{A} = \{a, b, c\}$, tenemos que $L_1 \setminus\!\setminus L_2 = \{\varepsilon, aa\varepsilon, c\varepsilon\}$. Si analizamos el lenguaje $L_1 \setminus\!\setminus L_2$, tenemos lo siguiente: la palabra ε pertenece a $L_1 \setminus\!\setminus L_2$ porque la palabra $a\varepsilon$ pertenece a L_2 y $\varepsilon \cdot a\varepsilon$ pertenece a L_1; la palabra $aa\varepsilon$ pertenece a $L_1 \setminus\!\setminus L_2$ porque la palabra $c\varepsilon$ pertenece a L_2 y $aa\varepsilon \cdot c\varepsilon$ pertenece a L_1; finalmente, la palabra $c\varepsilon$ pertenece a $L_1 \setminus\!\setminus L_2$ porque $c\varepsilon$ pertenece a L_2 y la palabra $c\varepsilon \cdot c\varepsilon$ pertenece a L_1. Por otro lado, $L_2 \setminus\!\setminus L_1 = \{\varepsilon\}$ porque ε pertenece a L_2 y la palabra $\varepsilon \cdot \varepsilon$ pertenece a L_1.

El tipo de la función que calcula el cociente es el siguiente:

$$2^{\mathbb{A}^*} \times 2^{\mathbb{A}^*} \to 2^{\mathbb{A}^*}$$

El tipo indica que, dados dos subconjuntos de $\mathbb{A}^*$, se obtiene otro subconjunto de $\mathbb{A}^*$. Dicho de otra forma, dados dos elementos de $2^{\mathbb{A}^*}$, se obtiene otro elemento de $2^{\mathbb{A}^*}$. Es decir, dados dos lenguajes, se obtiene otro lenguaje.

Si χ_{L_1} y χ_{L_2} son las funciones características de los lenguajes L_1 y L_2, respectivamente, entonces la función característica de $L_1 \setminus\!\setminus L_2$ puede ser expresada de la siguiente forma:

$$\chi_{L_1 \setminus\!\setminus L_2} : \mathbb{A}^* \to 2$$

$$\chi_{L_1 \setminus\!\setminus L_2}(w) = \begin{cases} 1 & \text{si } \exists u, v(u \in \mathbb{A}^* \wedge v \in \mathbb{A}^* \wedge (\chi_{L_1}(u) = 1) \wedge (\chi_{L_2}(v) = 1) \wedge (u = w \cdot v)) \\ 0 & \text{en caso contrario} \end{cases}$$

Propiedades del cociente:

- Para dos lenguajes cualesquiera L_1 y L_2, se cumple que si $\varepsilon \in L_2$, entonces $L_1 \subseteq (L_1 \setminus\!\setminus L_2)$.
- Para cualquier lenguaje L, se cumple $L \setminus\!\setminus \{\varepsilon\} = L$.
- Para cualquier lenguaje L, si $\varepsilon \in L$, entonces se cumple $\{\varepsilon\} \setminus\!\setminus L = \{\varepsilon\}$.
- Para cualquier lenguaje L, si $\varepsilon \notin L$, entonces se cumple $\{\varepsilon\} \setminus\!\setminus L = \varnothing$.
- Para cualquier lenguaje L, se cumple $L \setminus\!\setminus \varnothing = \varnothing$.

- Para cualquier lenguaje L, se cumple $\varnothing \setminus\setminus L = \varnothing$.
- Para cualquier lenguaje L, se cumple $\mathbb{A}^* \setminus\setminus L = \mathbb{A}^*$.
- Para cualquier lenguaje L, si $L \neq \varnothing$, entonces se cumple $L \setminus\setminus L \neq \varnothing$ y $\varepsilon \in L \setminus\setminus L$.
- En general, $L \setminus\setminus L \neq \{\varepsilon\}$.
- En general, $(L_1 \setminus\setminus L_2) \circ L_2 \neq L_1$.

3.4.

Conjuntos enumerables y no enumerables: el caso de $\mathbb{A}^*$ y $\mathbb{2}^{\mathbb{A}^*}$

3.4.1 Funciones biyectivas: funciones codificadoras

Sean Q y S dos conjuntos. Se dice que una función f de la forma $Q \to S$ es **biyectiva** si cumple las siguientes condiciones:

(a**)** Si k es un elemento de Q, entonces en S existe un elemento i que cumple $f(k) = i$.

(b**)** Si i es un elemento de S, entonces en Q existe un elemento k que cumple $f(k) = i$.

(c**)** Si k_1 y k_2 son dos elementos distintos de Q, entonces $f(k_1)$ y $f(k_2)$ son distintos.

Como consecuencia de cumplir esas tres condiciones, si i_1 e i_2 son dos elementos distintos de S, entonces existen dos elementos distintos k_1 y k_2 en Q que cumplen $f(k_1) = i_1$ y $f(k_2) = i_2$.

Las funciones que cumplen las condiciones *(a)* y *(b)* son **suprayectivas** y las funciones que cumplen las condiciones *(a)* y *(c)* son **inyectivas**.

Si existe una función biyectiva de la forma $Q \to S$, entonces Q y S tienen el mismo número de elementos (la misma cardinalidad) y se tiene una correspondencia uno-a-uno entre Q y S.

Si existe una función inyectiva de la forma $Q \to S$, entonces S tiene por lo menos tantos elementos como Q. En general, S podría tener más elementos que Q.

Si existe una función suprayectiva de la forma $Q \to S$, entonces Q tiene por lo menos tantos elementos como S. En general, Q podría tener más elementos que S.

Una función biyectiva f de la forma $Q \to S$, es una función **codificadora**. Es decir, sirve para codificar o representar elementos de Q mediante elementos de S. Dado un elemento $i \in Q$,

el elemento $f(i)$, perteneciente a S, es una codificación de i o una manera de representar i mediante un elemento de S.

Si una función f de la forma $Q \to S$ es biyectiva, entonces su función inversa $f^{-1} : S \to Q$ es también biyectiva y sirve para codificar o representar elementos de S mediante elementos de Q. La función inversa f^{-1} viene a ser la función **descodificadora** con respecto a f. De la misma forma, f puede ser entendida como la función descodificadora con respecto a f^{-1}.

3.4.2 Condición para que un conjunto sea enumerable

3.4.2.1 Criterio general

Un conjunto finito o infinito es enumerable (o contable) si sus elementos pueden ser colocados en una lista de tal forma que cada elemento sea alcanzable en un número finito de pasos empezando desde la primera posición (desde la izquierda). Para probar de manera formal o matemática que un conjunto C es enumerable, basta con probar alguna de las siguientes tres opciones:

- existe una función inyectiva de la forma $C \to \mathbb{N}$
- existe una función biyectiva de la forma $\mathbb{N} \to C$
- existe una función biyectiva que va desde un subconjunto de $\mathbb{N}$ a C.

Hay que recordar que el conjunto $\mathbb{N}$ es un subconjunto de sí mismo, es decir, de $\mathbb{N}$. Por otra parte, hay que tener clara la diferencia entre un conjunto y una lista: en un conjunto no hay orden y tampoco repeticiones (o las repeticiones no cuentan); en una lista se considera que los elementos están ordenados, teniendo en cuenta la posición que ocupan empezando desde la izquierda, y puede haber elementos repetidos. A continuación, se muestran algunos casos que ilustran esta diferencia entre conjuntos y listas. Los conjuntos y las listas de estos ejemplos, contienen palabras definidas sobre el alfabeto $\mathbb{A} = \{a, b, c\}$:

$$\{bcc\varepsilon, aaa\varepsilon, a\varepsilon, cb\varepsilon\} = \{aaa\varepsilon, a\varepsilon, bcc\varepsilon, cb\varepsilon\}$$
$$\{bcc\varepsilon, aaa\varepsilon, a\varepsilon, cb\varepsilon, aaa\varepsilon, cb\varepsilon, cb\varepsilon\} = \{aaa\varepsilon, a\varepsilon, bcc\varepsilon, cb\varepsilon\}$$

$$[bcc\varepsilon, aaa\varepsilon, a\varepsilon, cb\varepsilon] \neq [aaa\varepsilon, a\varepsilon, bcc\varepsilon, cb\varepsilon]$$
$$[bcc\varepsilon, aaa\varepsilon, a\varepsilon, cb\varepsilon, aaa\varepsilon, cb\varepsilon, cb\varepsilon] \neq [aaa\varepsilon, a\varepsilon, bcc\varepsilon, cb\varepsilon]$$

3.4.2.2 Conjuntos finitos: todos son enumerables

Todo conjunto finito es enumerable porque sus elementos se pueden poner en una lista y cualquier elemento es alcanzable en un número finito de pasos, empezando desde la izquierda. Por ejemplo, consideremos el alfabeto $\mathbb{A} = \{a, b, c\}$ y el conjunto finito D formado por las palabras de $\mathbb{A}^*$ que tienen longitud 2. Por tanto, $D = \{aa\varepsilon, ab\varepsilon, ac\varepsilon, ba\varepsilon, bb\varepsilon, bc\varepsilon, ca\varepsilon, cb\varepsilon, cc\varepsilon\}$. Los elementos del conjunto D se pueden poner en una lista, por ejemplo,

$$[\, aa\varepsilon, ab\varepsilon, ac\varepsilon, ba\varepsilon, bb\varepsilon, bc\varepsilon, ca\varepsilon, cb\varepsilon, cc\varepsilon \,]$$

y cualquier elemento es alcanzable en un número finito de pasos empezando desde la izquierda. Podemos considerar que $aa\varepsilon$ es alcanzable en cero pasos y, por consiguiente, $bc\varepsilon$ será alcanzable en cinco pasos, $ac\varepsilon$ en dos pasos, etc.

Formalmente, el conjunto D es enumerable porque existe una función biyectiva que va de $\{0, 1, \ldots, 8\}$, que es subconjunto de $\mathbb{N}$, al conjunto D:

$$\begin{aligned} &f : \{0, 1, \ldots, 8\} \to D \\ &f(0) = aa\varepsilon \\ &f(1) = ab\varepsilon \\ &f(2) = ac\varepsilon \\ &f(3) = ba\varepsilon \\ &f(4) = bb\varepsilon \\ &f(5) = bc\varepsilon \\ &f(6) = ca\varepsilon \\ &f(7) = cb\varepsilon \\ &f(8) = cc\varepsilon \end{aligned}$$

Además, hay otras posibilidades de poner los elementos del conjunto D en una lista, colocando los elementos en otro orden. Por ejemplo,

$$[\, bb\varepsilon, bc\varepsilon, aa\varepsilon, ba\varepsilon, cc\varepsilon, ab\varepsilon, ca\varepsilon, cb\varepsilon, ac\varepsilon \,]$$

o también

$$[\, ca\varepsilon, bc\varepsilon, bb\varepsilon, ac\varepsilon, cb\varepsilon, ba\varepsilon, ab\varepsilon, cc\varepsilon, aa\varepsilon \,]$$

etc.

En estos casos las funciones biyectivas correspondientes serían las siguientes:

$$\begin{aligned} &f : \{0, 1, \ldots, 8\} \to D \\ &f(0) = bb\varepsilon \\ &f(1) = bc\varepsilon \\ &f(2) = aa\varepsilon \\ &f(3) = ba\varepsilon \\ &f(4) = cc\varepsilon \\ &f(5) = ab\varepsilon \\ &f(6) = ca\varepsilon \\ &f(7) = cb\varepsilon \\ &f(8) = ac\varepsilon \end{aligned}$$

y

$$
\begin{aligned}
&f : \{0, 1, \ldots, 8\} \to D \\
&f(0) = ca\varepsilon \\
&f(1) = bc\varepsilon \\
&f(2) = bb\varepsilon \\
&f(3) = ac\varepsilon \\
&f(4) = cb\varepsilon \\
&f(5) = ba\varepsilon \\
&f(6) = ab\varepsilon \\
&f(7) = cc\varepsilon \\
&f(8) = aa\varepsilon
\end{aligned}
$$

Recordemos que en los conjuntos el orden no tiene importancia y, por tanto, el conjunto $\{ca\varepsilon, bc\varepsilon, bb\varepsilon\}$ y el conjunto $\{bc\varepsilon, bb\varepsilon, ca\varepsilon\}$ son el mismo conjunto. En cambio, en las listas el orden sí tiene importancia y, consecuentemente, la lista $[ca\varepsilon, bc\varepsilon, bb\varepsilon]$ y la lista $[bc\varepsilon, bb\varepsilon, ca\varepsilon]$ son distintas. Por otro lado, en los conjuntos los elementos repetidos no influyen y, por consiguiente, el conjunto $\{ca\varepsilon, bc\varepsilon, bb\varepsilon\}$ y el conjunto $\{\underline{ca\varepsilon}, bc\varepsilon, \underline{ca\varepsilon}, bb\varepsilon\}$ son el mismo conjunto. En el caso de las listas, los elementos repetidos sí influyen, lo cual conlleva que la lista $[ca\varepsilon, bc\varepsilon, bb\varepsilon]$ y la lista $[\underline{ca\varepsilon}, bc\varepsilon, \underline{ca\varepsilon}, bb\varepsilon]$ sean distintas.

3.4.2.3 Conjuntos infinitos: algunos son enumerables, otros no

En el caso de conjuntos infinitos, algunos son enumerables y otros no. Formalmente, un **conjunto infinito** C es **enumerable** (o contable) si existe una función $\mathbb{N} \to C$ que es biyectiva, es decir, a cada elemento diferente de $\mathbb{N}$ le corresponde un elemento diferente de C y a cada elemento diferente de C le corresponde un elemento diferente de $\mathbb{N}$. Esa función ordena los elementos de C: indica qué elemento va en la posición 0, qué elemento va en la posición 1, en la posición 2, etc. Otra opción es probar que existe una función injectiva de la forma $C \to \mathbb{N}$.

En el apartado 3.4.1, se ha indicado que las funciones biyectivas son también funciones codificadoras. A la hora de estudiar la computabilidad y la complejidad computacional, nos va a interesar el poder codificar unos tipos de elementos mediante otros tipos de elementos. Es por ello que al probar que algunos conjuntos infinitos son enumerables, vamos a centrarnos en funciones biyectivas que van de $\mathbb{N}$ a esos conjuntos.

En los apartados que vienen a continuación, se probará que algunos conjuntos infinitos como $\mathbb{Z}$, $\mathbb{N}^{\times k}$ con $k \in \mathbb{N}$ y $\mathbb{A}^*$ son enumerables, mientras que conjuntos infinitos como $\mathbb{R}$ y $2^{\mathbb{A}^*}$ no son enumerables.

3.4.2.4 Significado computacional de la enumerabilidad

El que los conjuntos $\mathbb{Z}$, $\mathbb{N}^{\times k}$ (para cualquier $k \in \mathbb{N}$) y $\mathbb{A}^*$ sean enumerables, quiere decir que el tamaño de esos conjuntos es igual al tamaño de $\mathbb{N}$. Dicho de otra forma, la infinitud de esos

conjuntos es igual a la infinitud de $\mathbb{N}$, aunque pueda parecer que no. Por otro lado, el que los conjuntos $\mathbb{R}$ y $2^{\mathbb{A}^*}$ no sean enumerables quiere decir que esos conjuntos contienen más elementos que el conjunto $\mathbb{N}$. Por consiguiente, aunque $\mathbb{R}$ y $\mathbb{N}$ sean conjuntos infinitos, su infinitud es distinta. De la misma forma, aunque $2^{\mathbb{A}^*}$ y $\mathbb{N}$ sean conjuntos infinitos, su infinitud es distinta.

Los conjuntos $\mathbb{A}^*$ y $2^{\mathbb{A}^*}$ son infinitos pero mientras que el primero es enumerable el segundo no lo es. Esto muestra que la infinitud de $\mathbb{A}^*$ y la infinitud de $2^{\mathbb{A}^*}$ son distintas. Desde el punto de vista de la computación, los conjuntos enumerables son mucho más manejables que los no enumerables.

Si un conjunto es enumerable, entonces existe un algoritmo capaz de ir generando una lista de todos los elementos de ese conjunto de tal forma que si elegimos cualquier elemento de ese conjunto, ese elemento aparecerá en la lista tras un número finito de pasos.

Si un conjunto no es enumerable, dado cualquier algoritmo ideado para generar la lista formada por los elementos de ese conjunto, siempre habrá algún elemento del conjunto que nunca será generado. Por tanto, no existe ningún algoritmo capaz de ir generando una lista de todos los elementos de ese conjunto de tal forma que para todo elemento de ese conjunto se pueda garantizar que antes o después aparecerá en la lista.

3.4.3 El conjunto infinito $\mathbb{P}\mathrm{ar}$ de los naturales pares es enumerable

El conjunto $\mathbb{P}\mathrm{ar}$ de los números naturales pares es enumerable. Una manera de poner sus elementos en una lista sería la siguiente:

$$[\,0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, \ldots\,]$$

Si consideramos que el 0 se alcanza en cero pasos, necesitamos cinco pasos para llegar al 10, seis para llegar al 12 y diez para llegar al 20. La cuestión es que dado cualquier número natural par k, ese número es alcanzable en una cantidad finita de pasos.

Para probar que $\mathbb{P}\mathrm{ar}$ es enumerable, se va a formular una función biyectiva que va de $\mathbb{N}$ a $\mathbb{P}\mathrm{ar}$. Puesto que esa función sirve para codificar o representar elementos de $\mathbb{N}$ mediante elementos de $\mathbb{P}\mathrm{ar}$, vamos a denominarla $Q_{\mathbb{N}\to\mathbb{P}\mathrm{ar}}$. Su inversa, $Q_{\mathbb{P}\mathrm{ar}\to\mathbb{N}}$, servirá para descodificar, o dicho de otra forma, para codificar elementos de $\mathbb{P}\mathrm{ar}$ mediante elementos de $\mathbb{N}$. El símbolo Q es la letra griega *Qoppa*. En la tabla 3.4.1 se muestra la especificación de la función $Q_{\mathbb{N}\to\mathbb{P}\mathrm{ar}}$ y en la tabla 3.4.2 se muestra la especificación de la función $Q_{\mathbb{P}\mathrm{ar}\to\mathbb{N}}$. Al ser esas dos funciones la una inversa de la otra, se cumplirá $Q_{\mathbb{P}\mathrm{ar}\to\mathbb{N}}(Q_{\mathbb{N}\to\mathbb{P}\mathrm{ar}}(k)) = k$ para cualquier $k \in \mathbb{N}$ y, también se cumplirá $Q_{\mathbb{N}\to\mathbb{P}\mathrm{ar}}(Q_{\mathbb{P}\mathrm{ar}\to\mathbb{N}}(j)) = j$ para cualquier $j \in \mathbb{P}\mathrm{ar}$.

De manera informal, la codificación que realiza la función $Q_{\mathbb{N}\to\mathbb{P}\mathrm{ar}}$ es la siguiente:

$$
\begin{aligned}
&\mathcal{Q}_{\mathbb{N}\to\mathbb{P}\mathrm{ar}}(0) = 0\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{P}\mathrm{ar}}(1) = 2\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{P}\mathrm{ar}}(2) = 4\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{P}\mathrm{ar}}(3) = 6\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{P}\mathrm{ar}}(4) = 8\\
&\vdots
\end{aligned}
$$

El que exista al menos una función biyectiva de la forma $\mathbb{N} \to \mathbb{P}\mathrm{ar}$, quiere decir que el tamaño de esos dos conjuntos es el mismo: por cada elemento de distinto de $\mathbb{P}\mathrm{ar}$ hay un elemento distinto en $\mathbb{N}$.

Es importante recalcar que no vale cualquier forma de listar los elementos del conjunto dado. Consideremos para los números naturales pares la lista en la que primero van todos los números naturales pares que son potencias enteras de 2 y luego todos los demás:

$$[\,2, 4, 8, 16, 32, 64, 128, \ldots, 0, 6, 10, 12, 14, 18, 20, 22, 24, \ldots\,]$$

Si consideramos que el 2 se alcanza en cero pasos, necesitamos, por ejemplo, tres pasos para llegar al 16 y cinco para llegar al 64. ¿Pero qué pasa con los números naturales pares que no son potencias enteras de 2? ¿Cuántos pasos necesitamos para llegar a 0? ¿Es 0 alcanzable? ¿Son los números naturales pares que no son potencias enteras de 2 alcanzables en esa lista? La respuesta es que no. Si partimos desde el 2 y vamos recorriendo la lista hacia la derecha, nunca llegaremos a los números naturales pares que no son potencias enteras de 2 porque hay infinitos números naturales pares que sí son potencias enteras de 2. Matemáticamente, no existe ninguna función biyectiva de la forma $\mathbb{N} \to \mathbb{P}\mathrm{ar}$ que establezca ese orden en $\mathbb{P}\mathrm{ar}$.

Por tanto, se ve que el orden en el que se listan los elementos de un conjunto infinito es importante a la hora de probar que el conjunto es enumerable. No vale cualquier manera de listar los elementos, hay que encontrar la manera adecuada de listarlos.

3.4.4 El conjunto infinito $\mathbb{Z}$ es enumerable

El conjunto $\mathbb{Z}$ de los números enteros es enumerable. Una manera de poner sus elementos en una lista sería la siguiente:

$$[\,0, 1, \text{-}1, 2, \text{-}2, 3, \text{-}3, 4, \text{-}4, 5, \text{-}5, 6, \text{-}6, 7, \text{-}7, 8, \text{-}8, 9, \text{-}9, 10, \text{-}10, 11, \text{-}11, 12, \text{-}12, \ldots\,]$$

Si consideramos que el 0 se alcanza en cero pasos, necesitamos cinco pasos para llegar al 3, seis para llegar al -3 y catorce al -7. La cuestión es que dado cualquier número k, ese número es alcanzable en una cantidad finita de pasos.

Para probar que $\mathbb{Z}$ es enumerable, se va a formular una función biyectiva que va de $\mathbb{N}$ a $\mathbb{Z}$. Puesto que esa función sirve para codificar o representar elementos de $\mathbb{N}$ mediante elementos de $\mathbb{Z}$, vamos a denominarla $\mathcal{Q}_{\mathbb{N}\to\mathbb{Z}}$. Su inversa, $\mathcal{Q}_{\mathbb{Z}\to\mathbb{N}}$, servirá para descodificar, o dicho de

otra forma, para codificar elementos de $\mathbb{Z}$ mediante elementos de $\mathbb{N}$. El símbolo $\mathcal{Q}$ es la letra griega *Qoppa*. En la tabla 3.4.3 se muestra la especificación de la función $\mathcal{Q}_{\mathbb{N}\to\mathbb{Z}}$ y en la tabla 3.4.4 se muestra la especificación de la función $\mathcal{Q}_{\mathbb{Z}\to\mathbb{N}}$. Al ser esas dos funciones la una inversa de la otra, se cumplirá $\mathcal{Q}_{\mathbb{Z}\to\mathbb{N}}(\mathcal{Q}_{\mathbb{N}\to\mathbb{Z}}(k)) = k$ para cualquier $k \in \mathbb{N}$ y, también se cumplirá $\mathcal{Q}_{\mathbb{N}\to\mathbb{Z}}(\mathcal{Q}_{\mathbb{Z}\to\mathbb{N}}(j)) = j$ para cualquier $j \in \mathbb{Z}$.

De manera informal, la codificación que realiza la función $\mathcal{Q}_{\mathbb{N}\to\mathbb{Z}}$ es la siguiente:

$$\begin{aligned}
&\mathcal{Q}_{\mathbb{N}\to\mathbb{Z}}(0) = 0\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{Z}}(1) = 1\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{Z}}(2) = \text{-}1\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{Z}}(3) = 2\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{Z}}(4) = \text{-}2\\
&\vdots
\end{aligned}$$

El que exista al menos una función biyectiva de la forma $\mathbb{N} \to \mathbb{Z}$, quiere decir que el tamaño de esos dos conjuntos es el mismo: por cada elemento de distinto de $\mathbb{Z}$ hay un elemento distinto en $\mathbb{N}$.

Es importante recalcar que no vale cualquier forma de listar los elementos del conjunto dado. Consideremos para los números enteros la lista en la que primero van todos los no negativos (es decir, el cero y los positivos) y luego todos los negativos:

$$[\,0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, \ldots, \text{-}1, \text{-}2, \text{-}3, \text{-}4, \text{-}5, \text{-}6, \text{-}7, \text{-}8, \text{-}9, \text{-}10, \text{-}11, \ldots\,]$$

Si consideramos que el 0 se alcanza en cero pasos, necesitamos, por ejemplo, tres pasos para llegar al 3 y seis para llegar al 6. ¿Pero qué pasa con los números negativos? ¿Cuántos pasos necesitamos para llegar a -1? ¿Es -1 alcanzable? ¿Son los números negativos alcanzables en esa lista? La respuesta es que no. Si partimos desde el 0 y vamos recorriendo la lista hacia la derecha, nunca llegaremos a los números negativos porque hay infinitos números positivos. Matemáticamente, no existe ninguna función biyectiva de la forma $\mathbb{N} \to \mathbb{Z}$ que establezca ese orden en $\mathbb{Z}$.

Por tanto, se ve que el orden en el que se listan los elementos de un conjunto infinito es importante a la hora de probar que el conjunto es enumerable. No vale cualquier manera de listar los elementos, hay que encontrar la manera adecuada de listarlos.

3.4.5 El conjunto $\mathbb{N}^{\times k}$ es enumerable para cualquier $k \in \mathbb{N}$

Mediante $\mathbb{N}^{\times k}$, denotamos el producto cartesiano $\mathbb{N} \times \mathbb{N} \times \cdots \times \mathbb{N}$ donde $\mathbb{N}$ aparece k veces. Se utilizará el formato $\mathbb{N}^{\times k}$ en vez del formato $\mathbb{N}^k$ —más habitual en matemáticas—, para diferenciar la operación de exponenciación de lenguajes con respecto a la concatenación (apartado 3.3.6.6) y la exponenciación de conjuntos con respecto al producto cartesiano. De esta forma, dado un lenguaje L, mediante L^3 denotamos la concatenación de lenguajes $L \circ L \circ L$, mientras

Especificación
$Q_{\mathbb{N}\to\mathbb{P}\mathrm{ar}} : \mathbb{N} \to \mathbb{P}\mathrm{ar}$ $Q_{\mathbb{N}\to\mathbb{P}\mathrm{ar}}(k) = 2 * k$

Tabla 3.4.1. Función codificadora de $\mathbb{N}$ a $\mathbb{P}\mathrm{ar}$.

Especificación
$Q_{\mathbb{P}\mathrm{ar}\to\mathbb{N}} : \mathbb{P}\mathrm{ar} \to \mathbb{N}$ $Q_{\mathbb{P}\mathrm{ar}\to\mathbb{N}}(k) = k \; div \; 2$ donde div representa la división entera entre números naturales.

Tabla 3.4.2. Función codificadora de $\mathbb{P}\mathrm{ar}$ a $\mathbb{N}$.

Especificación
$Q_{\mathbb{N}\to\mathbb{Z}} : \mathbb{N} \to \mathbb{Z}$ $Q_{\mathbb{N}\to\mathbb{Z}}(k) = \begin{cases} -(k \;\; div \;\; 2) & \text{si } k \text{ es par} \\ (k \;\; div \;\; 2) + 1 & \text{si } k \text{ es impar} \end{cases}$ donde div representa la división entera entre números naturales.

Tabla 3.4.3. Función codificadora de $\mathbb{N}$ a $\mathbb{Z}$.

Especificación
$Q_{\mathbb{Z}\to\mathbb{N}} : \mathbb{Z} \to \mathbb{N}$ $Q_{\mathbb{Z}\to\mathbb{N}}(k) = \begin{cases} -(k \;\; * \;\; 2) & \text{si } k \text{ es negativo o } k = 0 \\ (k \;\; * \;\; 2) - 1 & \text{si } k \text{ es positivo} \end{cases}$ donde $*$ representa el producto o la multiplicación entre números enteros.

Tabla 3.4.4. Función codificadora de $\mathbb{Z}$ a $\mathbb{N}$.

que mediante $L^{\times 3}$ denotamos el producto cartesiano $L \times L \times L$.

La definición general de $\mathbb{N}^{\times k}$, siendo k un número natural, es la siguiente:

$$\{(e_1, e_2, e_3, \ldots, e_k) \mid \forall i((1 \leq i \leq k) \rightarrow e_i \in \mathbb{N})\}$$

Las estructuras de la forma $(e_1, e_2, e_3, \ldots, e_k)$ reciben el nombre de "tupla de k elementos" o "k-tupla".

En vez de analizar la enumerabilidad de $\mathbb{N}^{\times k}$ en general, se analizarán los casos concretos $k = 0$, $k = 1$ y $k = 2$. El resto de casos, se derivan de una adaptación directa del caso $k = 2$.

3.4.5.1 El conjunto finito $\mathbb{N}^{\times 0}$ es enumerable

Cuando k es 0, se tiene solo la tupla vacía o la 0-tupla $()$. Es decir, $\mathbb{N}^{\times 0} = \{()\}$. Por tanto, $\mathbb{N}^{\times 0}$ es un conjunto finito que tiene un único elemento y es enumerable porque, por ejemplo, se tiene la función biyectiva $f : \{0\} \rightarrow \mathbb{N}^{\times 0}$ definida como $f(0) = ()$.

3.4.5.2 El conjunto infinito $\mathbb{N}^{\times 1}$ es enumerable

El conjunto $\mathbb{N}^{\times 1}$ se denota habitualmente como $\mathbb{N}$. Los elementos de $\mathbb{N}^{\times 1}$ son tuplas de tamaño 1 o 1-tuplas de la forma (j), donde j es un número natural, pero en vez de escribir (j) se escribirá simplemente j.

Puesto que en la definición de conjunto enumerable (apartado 3.4.2.1) aparece $\mathbb{N}$ como parte de esa definición, en este apartado vamos a utilizar la denominación $\mathbb{N}$ para referirnos a la $\mathbb{N}$ del apartado 3.4.2.1 y vamos a utilizar la denominación $\mathbb{N}^{\times 1}$ para referirnos a la $\mathbb{N}$ de este apartado. De esta forma, podremos diferenciar las dos versiones de $\mathbb{N}$. Pero también hay que insistir en que $\mathbb{N}$ y $\mathbb{N}^{\times 1}$ son el mismo conjunto.

En cualquier caso, es trivial probar que $\mathbb{N}^{\times 1}$ (o $\mathbb{N}$) es enumerable. La manera más natural de poner sus elementos en una lista sería la siguiente:

$$[\,0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, \ldots\,]$$

Si consideramos que, por estar accesible desde el principio, el número 0 se alcanza en cero pasos, necesitamos tres pasos para llegar al número 3, catorce pasos para llegar al número 14 y cuarenta y seis pasos para llegar al número 46. La cuestión es que, dado cualquier número j, ese número es alcanzable en j pasos.

En la tabla 3.4.5 se muestra la especificación de la función $\mathcal{Q}_{\mathbb{N} \rightarrow \mathbb{N}^{\times 1}}$ y en la tabla 3.4.6 se muestra la especificación de la función $\mathcal{Q}_{\mathbb{N}^{\times 1} \rightarrow \mathbb{N}}$. Al ser la una, inversa de la otra, se cumplirá $\mathcal{Q}_{\mathbb{N}^{\times 1} \rightarrow \mathbb{N}}(\mathcal{Q}_{\mathbb{N} \rightarrow \mathbb{N}^{\times 1}}(j)) = j$ para cualquier $j \in \mathbb{N}$ y, también se cumplirá $\mathcal{Q}_{\mathbb{N} \rightarrow \mathbb{N}^{\times 1}}(\mathcal{Q}_{\mathbb{N}^{\times 1} \rightarrow \mathbb{N}}(j)) = j$ para cualquier $j \in \mathbb{N}^{\times 1}$.

La función biyectiva $\mathcal{Q}_{\mathbb{N} \rightarrow \mathbb{N}^{\times 1}}$ viene a representar lo siguiente:

Especificación
$\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 1}} : \mathbb{N} \to \mathbb{N}^{\times 1}$ $\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 1}}(j) = j$

Tabla 3.4.5. Función codificadora de $\mathbb{N}$ a $\mathbb{N}^{\times 1}$.

Especificación
$\mathcal{Q}_{\mathbb{N}^{\times 1}\to\mathbb{N}} : \mathbb{N}^{\times 1} \to \mathbb{N}$ $\mathcal{Q}_{\mathbb{N}^{\times 1}\to\mathbb{N}}(j) = j$

Tabla 3.4.6. Función codificadora de $\mathbb{N}^{\times 1}$ a $\mathbb{N}$.

$$
\begin{aligned}
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 1}}(0) = 0\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 1}}(1) = 1\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 1}}(2) = 2\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 1}}(3) = 3\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 1}}(4) = 4\\
&\vdots
\end{aligned}
$$

Hay más maneras de listar los elementos de $\mathbb{N}^{\times 1}$. Una posibilidad de poner los elementos de $\mathbb{N}^{\times 1}$ en una lista es la siguiente:

$$[\,5, 0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, \ldots\,]$$

Si consideramos que el número 5 se alcanza en cero pasos (por estar accesible desde el principio), necesitamos cuatro pasos para llegar al número 3, catorce pasos para llegar al número 14 y cuarenta y seis pasos para llegar al número 46.

La especificación de la función biyectiva, o codificadora, correspondiente a esa lista, se puede formular de la siguiente manera:

$$
\begin{aligned}
&f : \mathbb{N} \to \mathbb{N}^{\times 1}\\
&f(k) = \begin{cases} 5 & \text{si } k = 0\\ k-1 & \text{si } 1 \leq k \leq 5\\ k & \text{si } k \geq 6 \end{cases}
\end{aligned}
$$

Esa función biyectiva viene a representar lo siguiente:

$$
\begin{aligned}
f(0) &= 5 \\
f(1) &= 0 \\
f(2) &= 1 \\
f(3) &= 2 \\
f(4) &= 3 \\
f(5) &= 4 \\
f(6) &= 6 \\
f(7) &= 7 \\
&\vdots
\end{aligned}
$$

La función inversa sería la siguiente:

$$
\begin{aligned}
&f^{-1} : \mathbb{N}^{\times 1} \to \mathbb{N} \\
&f^{-1}(k) = \begin{cases} 0 & \text{si } k = 5 \\ k+1 & \text{si } 0 \leq k \leq 4 \\ k & \text{si } k \geq 6 \end{cases}
\end{aligned}
$$

De manera informal:

$$
\begin{aligned}
f^{-1}(0) &= 1 \\
f^{-1}(1) &= 2 \\
f^{-1}(2) &= 3 \\
f^{-1}(3) &= 4 \\
f^{-1}(4) &= 5 \\
f^{-1}(5) &= 0 \\
f^{-1}(6) &= 6 \\
f^{-1}(7) &= 7 \\
&\vdots
\end{aligned}
$$

Es importante darse cuenta de que no vale cualquier forma de listar los elementos del conjunto dado. Consideremos para $\mathbb{N}^{\times 1}$, la lista en la que primero van todos los pares y luego todos los impares:

$$[\,0, 2, 4, 6, 8, 10, 12, 14, \ldots, 1, 3, 5, 7, 9, 11, 13, 15, \ldots\,]$$

Si consideramos que el 0 se alcanza en cero pasos, necesitamos, por ejemplo, tres pasos para llegar al 6 y seis para llegar al 12. Pero ¿qué pasa con los números impares? ¿Cuántos pasos necesitamos para llegar a 1? ¿Es 1 alcanzable? ¿Son los números impares alcanzables en esa lista? La respuesta es que no. Si partimos desde el 0 y vamos recorriendo la lista hacia la derecha, nunca llegaremos a los números impares porque hay infinitos números pares.

El caso de los números naturales muestra que el orden en el que se listan los elementos de un conjunto infinito es importante a la hora de probar que el conjunto es enumerable. No vale cualquier manera de listar los elementos, hay que encontrar la manera adecuada de listarlos.

3.4.5.3 El conjunto infinito $\mathbb{N}^{\times 2}$ es enumerable

El conjunto $\mathbb{N}^{\times 2}$ o lo que es lo mismo, el conjunto $\mathbb{N} \times \mathbb{N}$, formado por pares ordenados [1] de números naturales es enumerable.

En la tabla 3.4.7, se muestra una manera de poner sus elementos en una lista.

Una posible manera de listar u ordenar los elementos de $\mathbb{N}^{\times 2}$:
$[\ \underbrace{(0,0)}_{\text{suma = 0}}\ ,\underbrace{(0,1),(1,0)}_{\text{suma = 1}},\underbrace{(0,2),(1,1),(2,0)}_{\text{suma = 2}},\underbrace{(0,3),(1,2),(2,1),(3,0)}_{\text{suma = 3}},\ldots]$

Tabla 3.4.7. Lista para ordenar los elementos de $\mathbb{N}^{\times 2}$.

Si consideramos que el $(0,0)$ se alcanza en cero pasos, necesitamos cinco pasos para llegar al $(2,0)$, seis para llegar al $(0,3)$ y nueve al $(3,0)$. La cuestión es que dado cualquier par (m,n), ese par es alcanzable en un número finito de pasos. En esta lista, los pares se ordenan primero por la suma, y entre los que dan la misma suma, primero van los que tienen el primer componente más pequeño. Dicho de otra forma, si (m,n) y (p,q) son dos pares que suman lo mismo, es decir, si se cumple $m+n=p+q$, entonces (m,n) irá antes que (p,q) si se cumple $((m*10)+n) < ((p*10)+q)$.

Para probar que $\mathbb{N}^{\times 2}$ es enumerable, se va a formular una función biyectiva que va de $\mathbb{N}$ a $\mathbb{N}^{\times 2}$. Puesto que esa función sirve para codificar o representar elementos de $\mathbb{N}$ mediante elementos de $\mathbb{N}^{\times 2}$, vamos a denominarla $\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}$. Su inversa, $\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}$, servirá para descodificar, o dicho de otra forma, para codificar elementos de $\mathbb{N}^{\times 2}$ mediante elementos de $\mathbb{N}$. En la tabla 3.4.8 se muestra la especificación de la función $\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}$ y en la tabla 3.4.9 se muestra la especificación de la función $\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}$. Al ser cada una de estas dos funciones, inversa de la otra, se cumplirá $\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}(\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}(j)) = j$ para cualquier $j \in \mathbb{N}$ y, también se cumplirá $\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}(\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}((j,\ell)) = (j,\ell)$ para cualquier $(j,\ell) \in \mathbb{N}^{\times 2}$.

En la especificación de las funciones $\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}$ y $\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}$, se han utilizado las funciones auxiliares $\Omega_{\mathbb{N}\to\mathbb{N}^{\times 2}}$ y $\Omega_{\mathbb{N}^{\times 2}\to\mathbb{N}}$.

La función biyectiva $\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}$ representa lo siguiente:

[1] En los pares ordenados, el orden de los componentes es significativo. Por ejemplo, el par ordenado $(5,20)$ y el par ordenado $(20,5)$ son distintos.

$$
\begin{aligned}
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}(0) = (0,0)\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}(1) = (0,1)\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}(2) = (1,0)\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}(3) = (0,2)\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}(4) = (1,1)\\
&\vdots
\end{aligned}
$$

Especificación de la función que, dado un número j de $\mathbb{N}$, devuelve el j-ésimo par de $\mathbb{N}^{\times 2}$ teniendo en cuenta la lista de la tabla 3.4.7:

$$
\begin{aligned}
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}} : \mathbb{N} \to \mathbb{N}^{\times 2}\\
&\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}(j) = \Omega_{\mathbb{N}\to\mathbb{N}^{\times 2}}(j,(0,0))
\end{aligned}
$$

Especificación de la función que, dados un número x de $\mathbb{N}$ y un par (y,z) de $\mathbb{N}^{\times 2}$, devuelve el x-ésimo par de $\mathbb{N}^{\times 2}$ a partir del par (y,z):

$$
\begin{aligned}
&\Omega_{\mathbb{N}\to\mathbb{N}^{\times 2}} : \mathbb{N} \times \mathbb{N}^{\times 2} \to \mathbb{N}^{\times 2}\\
&\Omega_{\mathbb{N}\to\mathbb{N}^{\times 2}}(x,(y,z)) = \begin{cases} (y,z) & \text{si } x = 0\\ \Omega_{\mathbb{N}\to\mathbb{N}^{\times 2}}(x-1,(y+1,z-1)) & \text{si } x \neq 0 \wedge z \neq 0\\ \Omega_{\mathbb{N}\to\mathbb{N}^{\times 2}}(x-1,(0,y+1)) & \text{si } x \neq 0 \wedge z = 0 \end{cases}
\end{aligned}
$$

Tabla 3.4.8. Relación uno-a-uno entre números de $\mathbb{N}$ y pares de $\mathbb{N}^{\times 2}$.

La función biyectiva $\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}$ representa lo siguiente:

$$
\begin{aligned}
&\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}(0,0) = 0\\
&\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}(0,1) = 1\\
&\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}(1,0) = 2\\
&\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}(0,2) = 3\\
&\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}(1,1) = 4\\
&\vdots
\end{aligned}
$$

Otra manera alternativa de probar que $\mathbb{N}^{\times 2}$ es enumerable es definiendo la siguiente función biyectiva g de la forma $\mathbb{N}^{\times 2} \to \mathbb{N}$:

$$
\begin{aligned}
&g : \mathbb{N}^{\times 2} \to \mathbb{N}\\
&g((m,n)) = \frac{(m+n)(m+n+1)}{2} + m
\end{aligned}
$$

Esta función g viene a ser una especificación alternativa de la función biyectiva $\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}$ y, por tanto, es la inversa de $\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}$.

Pero también es posible dar una opción alternativa que no es inversa de $\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}$. Por ejemplo:

Especificación de la función que, dado un par ordenado (j, ℓ) de $\mathbb{N}^{\times 2}$, devuelve el número de $\mathbb{N}$ que indica la posición de (j, ℓ) en la lista de la tabla 3.4.7:
$\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}} : \mathbb{N}^{\times 2} \to \mathbb{N}$ $\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}((j, \ell)) = \boldsymbol{\Omega}_{\mathbb{N}^{\times 2}\to\mathbb{N}}(0, (j, \ell))$

Especificación de la función que, dado un número x de $\mathbb{N}$ y un par ordenado (y, z) de $\mathbb{N}^{\times 2}$, devuelve el número de $\mathbb{N}$ que indica la posición de (y, z) en la lista de la tabla 3.4.7 considerando que la primera posición es x:
$\boldsymbol{\Omega}_{\mathbb{N}^{\times 2}\to\mathbb{N}} : \mathbb{N} \times \mathbb{N}^{\times 2} \to \mathbb{N}$ $\boldsymbol{\Omega}_{\mathbb{N}^{\times 2}\to\mathbb{N}}(x, (y, z)) = \begin{cases} x & \text{si } (y, z) = (0, 0) \\ \boldsymbol{\Omega}_{\mathbb{N}^{\times 2}\to\mathbb{N}}(x + 1, (y - 1, z + 1)) & \text{si } (y, z) \neq (0, 0) \wedge y \neq 0 \\ \boldsymbol{\Omega}_{\mathbb{N}^{\times 2}\to\mathbb{N}}(x + 1, (z - 1, 0)) & \text{si } (y, z) \neq (0, 0) \wedge y = 0 \end{cases}$

Tabla 3.4.9. Relación uno-a-uno entre pares ordenados de $\mathbb{N}^{\times 2}$ y números de $\mathbb{N}$.

$$h : \mathbb{N}^{\times 2} \to \mathbb{N}$$
$$h((m, n)) = 2^m(2 * n + 1) - 1$$

La función h es inyectiva.

Una tercera opción alternativa es la de la función que, por ejemplo, dado el par ordenado $(42, 2013)$ considera el par ordenado en el que los dos componentes se expresan con la misma longitud, es decir, $(\underline{0042}, 2513)$ e intercala los dígitos obteniendo un número de $\mathbb{N}$: $2\underline{0}5\underline{0}1\underline{4}3\underline{2}$. No daremos la fórmula general.

Otra vez insistimos en que no vale cualquier forma de listar los elementos del conjunto dado. Consideremos para $\mathbb{N}^{\times 2}$, la lista en la que primero van todos los pares ordenados cuyo primer componente es 0, luego todos los pares ordenados cuyo primer componente es uno, etc.

$$[\,(0,0),(0,1),(0,2),(0,3),\ldots,(1,0),(1,1),(1,2),(1,3),\ldots,(2,0),(2,1),(2,2),(2,3),\ldots\,]$$

Si consideramos que el $(0, 0)$ se alcanza en cero pasos, necesitamos, por ejemplo, tres pasos para llegar al $(0, 3)$ y veinte para para llegar al $(0, 20)$. Pero ¿qué pasa con los pares ordenados cuyo primer componente no es 0? ¿Cuántos pasos necesitamos para llegar a $(1, 0)$? ¿Es $(1, 0)$ alcanzable? La respuesta es que no. Si partimos desde el $(0, 0)$ y vamos recorriendo la lista hacia la derecha, nunca llegaremos al par ordenado $(1, 0)$ ni a los posteriores.

Este caso también nos muestra que el orden en el que se listan los elementos de un conjunto infinito es importante a la hora de probar que el conjunto es enumerable. No vale cualquier manera de listar los elementos, hay que encontrar la manera adecuada de listarlos.

3.4.5.4 El conjunto infinito $\mathbb{N}^{\times k}$ es enumerable para cualquier $k \geq 3$

El conjunto $\mathbb{N}^{\times k}$ es enumerable para cualquier $k \geq 3$. Esto se puede demostrar formulando funciones biyectivas $\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times k}}$ y $\mathcal{Q}_{\mathbb{N}^{\times k}\to\mathbb{N}}$ definidas en base a una generalización de la idea presentada para $\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}$ y $\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}$.

3.4.6 $\mathbb{A}^*$ es enumerable

Dado cualquier alfabeto $\mathbb{A}$, el conjunto infinito $\mathbb{A}^*$ es enumerable.

Por ejemplo, si consideramos el alfabeto $\mathbb{A} = \{a, b\}$, entonces una manera de listar los elementos de $\mathbb{A}^*$ para probar que $\mathbb{A}^*$ es enumerable, es la mostrada en la tabla 3.4.10, donde los elementos están ordenados primero por longitud y dentro de los que tienen la misma longitud, por orden alfabético.

Palabras de $\mathbb{A}^*$ ordenadas:
$[\varepsilon, a\varepsilon, b\varepsilon, aa\varepsilon, ab\varepsilon, ba\varepsilon, bb\varepsilon, aaa\varepsilon, aab\varepsilon, aba\varepsilon, abb\varepsilon, baa\varepsilon, bab\varepsilon, bba\varepsilon, bbb\varepsilon, aaaa\varepsilon, \ldots]$

Tabla 3.4.10. Un manera posible de listar las palabras de $\mathbb{A}^*$, siendo $\mathbb{A} = \{a, b\}$.

Puesto que un alfabeto $\mathbb{A}$ es un conjunto, no hay ningún orden establecido entre sus elementos. Para poder generar una lista como la mostrada en la tabla 3.4.10, es necesario establecer un orden entre los elementos de $\mathbb{A}$. Establecer un orden significa definir una función biyectiva que va del conjunto $\{1, 2, 3, \ldots, |\mathbb{A}|\}$ al conjunto $\mathbb{A}$, donde $|\mathbb{A}|$ denota el número de elementos de $\mathbb{A}$, es decir, la cardinalidad de $\mathbb{A}$.

Por ejemplo, si $\mathbb{A} = \{a, b\}$, podemos definir una función biyectiva que va de $\{1, 2\}$ a $\{a, b\}$ que para 1 devuelve a y para 2 devuelve b. La lista de la tabla 3.4.10 se ha obtenido teniendo en cuenta este orden de $\mathbb{A}$.

Para probar que $\mathbb{A}^*$ es enumerable, se va a formular una función biyectiva que va de $\mathbb{N}$ a $\mathbb{A}^*$. Puesto que esa función sirve para codificar o representar elementos de $\mathbb{N}$ mediante elementos de $\mathbb{A}^*$, vamos a denominarla $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}$. Su inversa, $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}$, servirá para descodificar, o dicho de otra forma, para codificar elementos de $\mathbb{A}^*$ mediante elementos de $\mathbb{N}$. En la tabla 3.4.11 se muestra la especificación de la función $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}$ y en la tabla 3.4.12 se muestra la especificación de la función $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}$. Al ser cada una de estas dos funciones, inversa de la otra, se cumplirá $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}(\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}(j)) = j$ para cualquier $j \in \mathbb{N}$ y, también se cumplirá $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}(\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}(w)) = w$ para cualquier $w \in \mathbb{A}^*$. En la especificación de las funciones $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}$ y $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}$, se han utilizado las funciones auxiliares $\Omega_{\mathbb{N}\to\mathbb{A}^*}$ y $\Omega_{\mathbb{A}^*\to\mathbb{N}}$.

Especificación de la función biyectiva que asocia a cada número de $\mathbb{N}$ una palabra de $\mathbb{A}^*$. La función es dependiente del orden establecido en el alfabeto $\mathbb{A}$ mediante una función biyectiva ord:

$$\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*} : \mathbb{N} \to A^*$$
$$\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}(k) = \mathcal{\Omega}_{\mathbb{N}\to\mathbb{A}^*}(ord, k, \varepsilon)$$

Especificación de la función que, dados un orden ord en $\mathbb{A}$, un número natural k y una palabra w de $\mathbb{A}^*$, devuelve la k-ésima palabra a partir de w:

$$\mathcal{\Omega}_{\mathbb{N}\to\mathbb{A}^*} : (\{x \mid x \in \mathbb{N} \wedge x \geq 1 \wedge x \leq |\mathbb{A}|\} \to \mathbb{A}) \times \mathbb{N} \times \mathbb{A}^* \to \mathbb{A}^*$$
$$\mathcal{\Omega}_{\mathbb{N}\to\mathbb{A}^*}(ord, k, w) = \begin{cases} w & \text{si } k = 0 \\ \mathcal{\Omega}_{\mathbb{N}\to\mathbb{A}^*}(ord, k-1, \mathsf{sig}(ord, w)) & \text{si } k \neq 0 \end{cases}$$

Especificación de la función que, dados un orden ord en $\mathbb{A}$ y una palabra w de $\mathbb{A}^*$, devuelve la siguiente palabra a partir de w:

$$\mathsf{sig} : (\{x \mid x \in \mathbb{N} \wedge x \geq 1 \wedge x \leq |\mathbb{A}|\} \to \mathbb{A}) \times \mathbb{A}^* \to \mathbb{A}^*$$
$$\mathsf{sig}(ord, w) = \begin{cases} ord(1)\varepsilon & \text{si } w = \varepsilon \\ w(1, |w|-1) \cdot (ord(ord^{-1}(w(|w|)) + 1)\varepsilon) & \text{si } w \neq \varepsilon \wedge w(|w|) \neq ord(|\mathbb{A}|) \\ \mathsf{sig}(ord, w(1, |w|-1)) \cdot (ord(1)\varepsilon) & \text{si } w \neq \varepsilon \wedge w(|w|) = ord(|\mathbb{A}|) \end{cases}$$

$|\mathbb{A}|$ denota el número de elementos del conjunto $\mathbb{A}$, es decir, la cardinalidad del conjunto $\mathbb{A}$. En un conjunto no hay ni repeticiones ni orden entre elementos.

ε denota la palabra vacía.
$\cdot$ denota la concatenación entre palabras.
$|w|$ denota la longitud de la palabra w.

ord denota cualquier función biyectiva entre el conjunto $\{1, \ldots, |\mathbb{A}|\}$ y $\mathbb{A}$. Por tanto, establece un orden entre los símbolos de $\mathbb{A}$.
ord^{-1} denota la función inversa de ord. Es, por tanto, una función biyectiva entre el conjunto $\mathbb{A}$ y $\{1, \ldots, |\mathbb{A}|\}$.

Tabla 3.4.11. Relación uno-a-uno entre números de $\mathbb{N}$ y palabras de $\mathbb{A}^*$.

Función biyectiva que asocia a cada palabra de $\mathbb{A}^*$ un número de $\mathbb{N}$.
La función es dependiente del orden establecido en el alfabeto $\mathbb{A}$ mediante una función biyectiva ord:

$$\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}} : A^* \to \mathbb{N}$$
$$\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}(w) = \boldsymbol{\Omega}_{\mathbb{A}^*\to\mathbb{N}}(ord, 0, w)$$

Especificación de la función que, dados un orden ord en $\mathbb{A}$, un número natural k y una palabra w de $\mathbb{A}^*$, devuelve la posición de w en la lista de palabras más k:

$$\boldsymbol{\Omega}_{\mathbb{A}^*\to\mathbb{N}} : (\{x \mid x \in \mathbb{N} \wedge x \geq 1 \wedge x \leq |\mathbb{A}|\} \to \mathbb{A}) \times \mathbb{N} \times \mathbb{A}^* \to \mathbb{A}^*$$
$$\boldsymbol{\Omega}_{\mathbb{A}^*\to\mathbb{N}}(ord, k, w) = \begin{cases} k & \text{si } w = \varepsilon \\ \boldsymbol{\Omega}_{\mathbb{A}^*\to\mathbb{N}}(ord, k+1, \mathsf{ant}(ord, w)) & \text{si } w \neq \varepsilon \end{cases}$$

Especificación de la función que, dados un orden en $\mathbb{A}$ y una palabra $w \neq \varepsilon$, devuelve la palabra anterior a partir de w:

$$\mathsf{ant} : (\{x \mid x \in \mathbb{N} \wedge x \geq 1 \wedge x \leq |\mathbb{A}|\} \to \mathbb{A}) \times \mathbb{A}^* \to \mathbb{A}^*$$
$$\mathsf{ant}(ord, w) = \begin{cases} \varepsilon & \text{si } w = ord(1)\varepsilon \\ w(1, |w|-1) \cdot (ord(ord^{-1}(w(|w|)) - 1)\varepsilon) & \text{si } w \neq ord(1)\varepsilon \wedge w(|w|) \neq ord(1) \\ \mathsf{ant}(ord, w(1, |w|-1)) \cdot (ord(|A|)\varepsilon) & \text{si } w \neq ord(1)\varepsilon \wedge w(|w|) = ord(1) \end{cases}$$

$|\mathbb{A}|$ denota el número de elementos del conjunto $\mathbb{A}$, es decir, la cardinalidad del conjunto $\mathbb{A}$. En un conjunto no hay ni repeticiones ni orden entre elementos.

ε denota la palabra vacía.
$\cdot$ denota la concatenación entre palabras.
$|w|$ denota la longitud de la palabra w.

ord denota cualquier función biyectiva entre el conjunto $\{1, \ldots, |\mathbb{A}|\}$ y $\mathbb{A}$.
Por tanto, establece un orden entre los símbolos de $\mathbb{A}$.
ord^{-1} denota la función inversa de ord. Es, por tanto, una función biyectiva entre el conjunto $\mathbb{A}$ y $\{1, \ldots, |\mathbb{A}|\}$.

Tabla 3.4.12. Relación uno-a-uno entre palabras de $\mathbb{A}^*$ y números de $\mathbb{N}$.

La función $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}$, dado un número entero k nos indica qué elemento de $\mathbb{A}^*$ se alcanza en k pasos. El número de pasos correspondiente a cada palabra depende del orden considerado entre los símbolos pertenecientes al alfabeto $\mathbb{A}$.

En el caso concreto $\mathbb{A} = \{a, b\}$, $ord(1) = a$ y $ord(2) = b$, la función biyectiva $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}$ asignará un número a cada palabra siguiendo el orden de la lista mostrada en la tabla 3.4.10, tal como se muestra en la tabla 3.4.13.

Palabras de $\mathbb{A}^*$ ordenadas:
$\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}(0) = \varepsilon$ $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}(1) = a\varepsilon$ $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}(2) = b\varepsilon$ $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}(3) = aa\varepsilon$ $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}(4) = ab\varepsilon$ $\vdots$

Tabla 3.4.13. Función biyectiva correspondiente a la lista de la tabla 3.4.10.

Tal como se ha mencionado para el caso concreto $\mathbb{A} = \{a, b\}$, $ord(1) = a$ y $ord(2) = b$, la cuestión es poder establecer un orden entre los elementos del conjunto $\mathbb{A}^*$. Dada una palabra w que no contiene ninguna aparición de a y tal que $|w| = k$, la siguiente palabra es la palabra x que solo contiene a-s y $|x| = k + 1$. En cambio, si w contiene alguna aparición de a y $|w| = k$, la siguiente palabra es la palabra x que tiene la misma longitud que w, es decir, $|x| = k$ y es alfabéticamente mayor. Así, como $bb\varepsilon$ no contiene ninguna aparición de a, la siguiente palabra es $aaa\varepsilon$ (todo a-s pero un elemento más). Por otro lado, como $baab\varepsilon$ sí contiene al menos una aparición de a, la siguiente palabra es $babb\varepsilon$, es decir, misma longitud pero alfabéticamente mayor (se sustituye por b la a que está más a la derecha). Esta idea sirve para cualquier alfabeto. Por tanto, dado cualquier alfabeto $\mathbb{A}$, el conjunto $\mathbb{A}^*$ es enumerable.

Para definir $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}$, se ha utilizado la función $\Omega_{\mathbb{N}\to\mathbb{A}^*}$. Esta función auxiliar $\Omega_{\mathbb{N}\to\mathbb{A}^*}$, recibe una función biyectiva concreta ord que establece un orden entre los símbolos de $\mathbb{A}$, un natural k y una palabra w y devuelve la k-ésima palabra a partir de w teniendo en cuenta el orden establecido por ord. Por otra parte, para dar la especificación de $\Omega_{\mathbb{N}\to\mathbb{A}^*}$, se ha utilizado la función sig, que dado un orden establecido en $\mathbb{A}$ y una palabra de $\mathbb{A}^*$, calcula la siguiente palabra. También se ha utilizado la función ord^{-1} que es la inversa de ord.

La función $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}$, dada una palabra w de $\mathbb{A}^*$, nos indica cuántos pasos se necesitan para llegar a w. El número de pasos correspondiente a cada palabra depende del orden considerado entre los símbolos pertenecientes al alfabeto $\mathbb{A}$.

En el caso concreto $\mathbb{A} = \{a, b\}$, $ord(1) = a$ y $ord(2) = b$, la función biyectiva $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}$ asociará a cada palabra de $\mathbb{A}^*$, un número natural siguiendo el orden de la lista mostrada en la tabla 3.4.10, tal como se muestra en la tabla 3.4.14.

Posición correspondiente a las palabras de $\mathbb{A}^*$:
$\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}(\varepsilon) = 0$ $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}(a\varepsilon) = 1$ $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}(b\varepsilon) = 2$ $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}(aa\varepsilon) = 3$ $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}(ab\varepsilon) = 4$ $\vdots$

Tabla 3.4.14. Función biyectiva inversa correspondiente a la lista de la tabla 3.4.10.

Para definir $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}$, se ha utilizado la función $\Omega_{\mathbb{A}^*\to\mathbb{N}}$. Esta función auxiliar $\Omega_{\mathbb{A}^*\to\mathbb{N}}$, recibe una función biyectiva concreta ord que establece un orden entre los símbolos de $\mathbb{A}$, un natural k y una palabra w y devuelve la suma de k y la posición correspondiente a la palabra w teniendo en cuenta el orden establecido por ord. Por otra parte, para dar la especificación de $\Omega_{\mathbb{A}^*\to\mathbb{N}}$, se ha utilizado la función ant, que dado un orden establecido en $\mathbb{A}$ y una palabra de $\mathbb{A}^*$, calcula la palabra anterior. También se ha utilizado la función ord^{-1} que es la inversa de ord.

Para probar que $\mathbb{A}^*$ es enumerable, otra opción es definir una función inyectiva $g : \mathbb{A}^* \to \mathbb{N}$. Para dar la definición de g, se necesita la función biyectiva $s : \mathbb{A} \to \{1, 2 \ldots, |\mathbb{A}|\}$ que, si $\mathbb{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, asocia un número $s(\alpha_j)$ a cada símbolo α_j de $\mathbb{A}$. Por ejemplo, $s(\alpha_j) = j$. La función g se define de la siguiente forma:

$$\begin{aligned} &g : \mathbb{A}^* \to \mathbb{N} \\ &g(\varepsilon) = 0 \\ &g(\alpha w) = s(\alpha) \times k^{|w|} + g(w) \end{aligned}$$

donde k es el número de símbolos de $\mathbb{A}$, ε es la palabra vacía, α es un símbolo de $\mathbb{A}$ y w es una palabra de $\mathbb{A}^*$.

Por ejemplo, si $\mathbb{A} = \{a, b\}$, entonces tenemos que $k = 2$, $\alpha_1 = a$, $\alpha_2 = b$, $s(a) = 1$, $s(b) = 2$ y

$$\begin{aligned} g(aab\varepsilon) &= 1 \times 2^2 + g(ab\varepsilon) \\ &= 1 \times 2^2 + 1 \times 2^1 + g(b\varepsilon) \\ &= 1 \times 2^2 + 1 \times 2^1 + 2 \times 2^0 + g(\varepsilon) \\ &= 4 + 2 + 2 + 0 \\ &= 8 \end{aligned}$$

Consideremos ahora otra manera de poner en una lista los elementos de $\mathbb{A}^*$:

$$[\,\varepsilon, a\varepsilon, aa\varepsilon, aaa\varepsilon, aaaa\varepsilon, aaaaa\varepsilon, \ldots\,]$$

donde primero van todas las palabras que no contienen ninguna aparición de b. Esta lista no sirve para probar que $\mathbb{A}^*$ es enumerable. El problema es que en esa lista, las palabras que contengan al menos una aparición de b son inalcanzables. Esto muestra que cualquier manera de listar los elementos no vale, hay que buscar una manera adecuada de poner los elementos en una lista.

Los conjuntos infinitos enumerables tienen el mismo tamaño que $\mathbb{N}$.

3.4.7 $2^{\mathbb{A}^*}$ no es enumerable

El conjunto $2^{\mathbb{A}^*}$ no es enumerable. No se puede definir ninguna función biyectiva de la forma $\mathbb{N} \to 2^{\mathbb{A}^*}$.

La prueba es por contradicción. Supondremos que $2^{\mathbb{A}^*}$ es enumerable y basándonos en esa suposición generaremos una contradicción. Por tanto, como el suponer que $2^{\mathbb{A}^*}$ es enumerable nos conducirá a una situación incoherente, deduciremos que $2^{\mathbb{A}^*}$ no es enumerable.

Por un lado, supongamos que $2^{\mathbb{A}^*}$ es enumerable y que, por consiguiente, existe una función biyectiva $g : \mathbb{N} \to 2^{\mathbb{A}^*}$. El que exista dicha función g significa que, por medio de g, a cada número natural distinto le corresponde un lenguaje distinto de $2^{\mathbb{A}^*}$ y a cada lenguaje distinto de $2^{\mathbb{A}^*}$ le corresponde un número distinto de $\mathbb{N}$. Si existe g, por medio de g podemos establecer una relación de uno a uno entre lenguajes y números naturales. De esta forma $g(0)$ sería un lenguaje, $g(1)$ sería otro lenguaje, $g(2)$ sería otro lenguaje, $g(3)$ sería otro lenguaje, etc. Recordemos que los elementos de $2^{\mathbb{A}^*}$ son subconjuntos de $\mathbb{A}^*$. Por tanto, $g(0)$ sería un subconjunto de $\mathbb{A}^*$, $g(1)$ sería otro subconjunto de $\mathbb{A}^*$, $g(2)$ sería otro subconjunto de $\mathbb{A}^*$, etc. Si existe g, podremos poner los elementos de $2^{\mathbb{A}^*}$ en una lista utilizando g:

$$[\,g(0), g(1), g(2), g(3), \ldots, g(j), \ldots\,]$$

Por otro lado, como sabemos que $\mathbb{A}^*$ es enumerable, sabemos que existe una función biyectiva $f : \mathbb{N} \to \mathbb{A}^*$ que asocia a cada número distinto de $\mathbb{N}$ una palabra distinta de $\mathbb{A}^*$ y asocia a cada palabra distinta de $\mathbb{A}^*$ un número distinto de $\mathbb{N}$. Por tanto, $f(0)$ es una palabra de $\mathbb{A}^*$, $f(1)$ es otra palabra de $\mathbb{A}^*$, $f(2)$ es otra palabra de $\mathbb{A}^*$, $f(3)$ es otra palabra de $\mathbb{A}^*$, etc. Utilizando f, podemos poner los elementos de $\mathbb{A}^*$ en una lista:

$$[\,f(0), f(1), f(2), f(3), \ldots, f(j), \ldots\,]$$

Resumiendo lo anterior, tenemos que $g(0), g(1), g(2), g(3), \ldots$ son lenguajes (elementos del conjunto $2^{\mathbb{A}^*}$) y $f(0), f(1), f(2), f(3), \ldots$ son palabras (elementos del conjunto $\mathbb{A}^*$).

Ahora, utilizando las funciones g y f vamos a construir un conjunto C "imposible" o contradictorio que debería pertenecer a $2^{\mathbb{A}^*}$.

La construcción de C es como sigue: para cada $k \in \mathbb{N}$, si la palabra $f(k)$ no pertenece al lenguaje $g(k)$, entonces hacemos que la palabra $f(k)$ esté en C. En cambio, si la palabra $f(k)$ pertenece al lenguaje $g(k)$, entonces hacemos que la palabra $f(k)$ no esté en C. Es decir, si la palabra $f(0)$ no está en el lenguaje $g(0)$, entonces la palabra $f(0)$ está en C y si la palabra $f(0)$ está en el lenguaje $g(0)$, entonces la palabra $f(0)$ no está en C. Si la palabra $f(1)$ no está en el lenguaje $g(1)$, entonces la palabra $f(1)$ está en C y si la palabra $f(1)$ está en el lenguaje $g(1)$, entonces la palabra $f(1)$ no está en C. Si la palabra $f(2)$ no está en el lenguaje $g(2)$, entonces la palabra $f(2)$ está en C y si la palabra $f(2)$ está en el lenguaje $g(2)$, entonces la palabra $f(2)$ no está en C. Y así con todos los números naturales. Consecuentemente, para el conjunto C se cumple lo siguiente:

$$\forall k((k \in \mathbb{N} \wedge f(k) \in g(k)) \rightarrow f(k) \notin C)$$
$$\forall k((k \in \mathbb{N} \wedge f(k) \notin g(k)) \rightarrow f(k) \in C)$$

El conjunto C construido de esta forma es un conjunto de palabras y, por tanto, pertenece a $2^{\mathbb{A}^*}$. Como g asocia un número natural a cada elemento de $2^{\mathbb{A}^*}$, g también asociará un número natural a C. Supongamos que ese número es j. Por tanto $g(j) = C$. El problema surge cuando consideramos la palabra $f(j)$ e intentamos averiguar si $f(j)$ pertenece al lenguaje C, es decir, al lenguaje $g(j)$. Por definición de C, si la palabra $f(j)$ pertenece al lenguaje $g(j)$, entonces la palabra $f(j)$ no pertenece a C y si la palabra $f(j)$ no pertenece al lenguaje $g(j)$, entonces la palabra $f(j)$ pertenece a C. Como C es $g(j)$, tenemos que si la palabra $f(j)$ pertenece a C, entonces la palabra $f(j)$ no pertenece a C y si la palabra $f(j)$ no pertenece a C, entonces la palabra $f(j)$ pertenece a C. Y esto es una contradicción porque no puede ser que una palabra pertenezca a C y a la vez no pertenezca a C. Esquemáticamente, suponiendo que $C = g(j)$ tenemos lo siguiente:

$$f(j) \in g(j) \rightarrow f(j) \notin C \qquad \text{¡contradicción! porque } C = g(j)$$
$$f(j) \notin g(j) \rightarrow f(j) \in C \qquad \text{¡contradicción! porque } C = g(j)$$

Como suponiendo que existe una función biyectiva $g : \mathbb{N} \rightarrow 2^{\mathbb{A}^*}$ hemos llegado a una contradicción, deducimos que g no existe y por tanto $2^{\mathbb{A}^*}$ no es enumerable.

3.4.8 $\mathbb{R}$ no es enumerable

Con el objetivo de que se pueda entender mejor la técnica utilizada para probar que el conjunto $2^{\mathbb{A}^*}$ no es enumerable, se va a probrar que el conjunto $\mathbb{R}$ no es enumerable haciendo uso de la misma técnica.

Para demostrar que $\mathbb{R}$ no es enumerable, se probará que el conjunto infinito $[0..1)$ no es enumerable. Como $[0..1)$ es un subconjunto de $\mathbb{R}$, si $[0..1)$ no es enumerable, entonces $\mathbb{R}$ tampoco

será enumerable. [2] La prueba de que $[0..1)$ no es enumerable consistirá en llegar a la conclusión de que no existe ninguna función biyectiva de la forma $\mathbb{N} \to [0..1)$.

Haciendo uso de la técnica de la contradicción, supondremos que $[0..1)$ es enumerable y basándonos en esa suposición generaremos una contradicción. Por tanto, como el suponer que $[0..1)$ es enumerable nos conducirá a una situación incoherente, deduciremos que $[0..1)$ no es enumerable.

Supongamos que $[0..1)$ es enumerable. Si [0..1) es enumerable, entonces existirá una función biyectiva $h : \mathbb{N} \to [0..1)$. Por medio de h, a cada número natural le corresponde un número real de $[0..1)$ y a cada número real de $[0..1)$ le corresponde un número natural de $\mathbb{N}$. Si existe h, por medio de h podemos establecer una relación de uno a uno entre los números de $[0..1)$ y los números naturales. De esta forma $h(0)$ sería un número de $[0..1)$, $h(1)$ sería otro número de $[0..1)$, $h(2)$ sería otro número de $[0..1)$, $h(3)$ sería otro número de $[0..1)$, etc. En definitiva, si existe h, podremos poner los elementos de $[0..1)$ en una lista utilizando h:

$$[\, h(0), h(1), h(2), h(3), \ldots, h(j), \ldots]$$

Por otro lado, sabemos que todo número del conjunto $[0..1)$ es de la forma "cero coma y una cantidad infinita de dígitos". Por ejemplo, el número cero es de la forma $0,00000\ldots$, con infinitos dígitos (donde cada uno de esos dígitos es 0). Esto quiere decir que los elementos de la lista $[\, h(0), h(1), h(2), h(3), \ldots, h(j), \ldots]$ tienen la siguiente forma:

$$h(0) = 0, d_0^0\, d_0^1\, d_0^2\, \ldots\, d_0^j\, \ldots$$

$$h(1) = 0, d_1^0\, d_1^1\, d_1^2\, \ldots\, d_1^j\, \ldots$$

$$h(2) = 0, d_2^0\, d_2^1\, d_2^2\, \ldots\, d_2^j\, \ldots$$

$$\vdots$$

$$h(j) = 0, d_j^0\, d_j^1\, d_j^2\, \ldots\, d_j^j\, \ldots$$

$$\vdots$$

donde cada componente d_m^k pertenece al conjunto $\{0, 1, 2, 3, \ldots, 9\}$.

Ahora, utilizando la función h vamos a construir un número q "imposible" o contradictorio que pertenece a $[0..1)$. Por pertenecer a $[0..1)$, el número q tendrá la siguiente forma:

$$q = 0, q_0\, q_1\, q_2\, \ldots\, q_j\, \ldots$$

[2] En el conjunto $[0..1)$ están el cero y todos los números reales de la forma "cero coma algo", pero no está el número 1.

donde cada componente q_i pertenece al conjunto $\{0, 1, 2, 3, \ldots, 9\}$.

La construcción de q es como sigue: para cada $\ell \in \mathbb{N}$, si el dígito d_ℓ^ℓ del número $h(\ell)$ es cero, entonces el ℓ-ésimo dígito de q, es decir, q_ℓ, será 1. En cambio, si el dígito d_ℓ^ℓ del número $h(\ell)$ no es cero, entonces el ℓ-ésimo dígito de q, es decir, q_ℓ, será 0. Es decir, si el dígito d_0^0 del número $h(0)$ es cero, entonces q_0 será 1 y si el dígito d_0^0 del número $h(0)$ no es cero, entonces q_0 será 0. De la misma forma, si el dígito d_1^1 del número $h(1)$ es cero, entonces q_1 será 1 y si el dígito d_1^1 del número $h(1)$ no es cero, entonces q_1 será 0. También de la misma forma, si el dígito d_2^2 del número $h(2)$ es cero, entonces q_2 será 1 y si el dígito d_2^2 del número $h(2)$ no es cero, entonces q_2 será 0. Y así con todos los números naturales. Cosecuentemente, para el número q se cumple lo siguiente:

$$\forall \ell((\ell \in \mathbb{N} \wedge d_\ell^\ell = 0) \rightarrow q_\ell = 1)$$
$$\forall \ell((\ell \in \mathbb{N} \wedge d_\ell^\ell \neq 0) \rightarrow q_\ell = 0)$$

El número q construido de esta forma pertenece al conjunto $[0..1)$. Como h asocia un número natural a cada elemento de $[0..1)$, h también asociará un número natural a q. Supongamos que ese número es j. Por tanto $h(j) = q$. Por construcción, todos los dígitos de q, es decir, todos los dígitos de $h(j)$, son ceros o unos. El problema surge cuando consideramos el número q e intentamos averiguar si el dígito q_j es 0 o 1. Por definición de q, si el dígito d_j^j del número $h(j)$ es cero, entonces el dígito q_j de q será 1 y si el dígito d_j^j del número $h(j)$ no es cero, entonces el dígito q_j de q será 0. Como q es $h(j)$, en primer lugar tenemos que $d_j^0 = q_0, d_j^1 = q_1, d_j^2 = q_2, \ldots, d_j^j = q_j, \ldots$. En segundo lugar tenemos que si el dígito d_j^j del número $h(j)$ es cero, entonces el dígito q_j de q (es decir, el dígito d_j^j) será 1 y si el dígito d_j^j del número $h(j)$ no es cero, entonces el dígito q_j de q (es decir, el dígito d_j^j) será 0. Y esto es una contradicción porque un dígito, en concreto d_j^j, no puede ser a la vez 0 y 1 y no puede ser a la vez distinto de 0 y 0. Esquemáticamente, suponiendo que $q = h(j)$ tenemos lo siguiente:

$$d_j^j = 0 \rightarrow q_j = 1 \qquad \text{¡contradicción! porque } q_j = d_j^j$$
$$d_j^j \neq 0 \rightarrow q_j = 0 \qquad \text{¡contradicción! porque } q_j = d_j^j$$

Como suponiendo que existe una función biyectiva $h : \mathbb{N} \rightarrow [0..1)$ hemos llegado a una contradicción, deducimos que h no existe y por tanto $[0..1)$ no es enumerable. Por último, deducimos que como $[0..1)$ no es enumerable, $\mathbb{R}$ tampoco es enumerable.

3.5.

Toda la información se puede expresar utilizando 0 y 1

3.5.1 Codificación de programas y datos mediante números naturales

En los apartados 3.4.4, 3.4.5 y 3.4.6, hemos definido varias funciones codificadoras. De ellas, las más importantes para nosotros son $Q_{\mathbb{N}\to\mathbb{N}^{\times 2}}$, $Q_{\mathbb{N}^{\times 2}\to\mathbb{N}}$, $Q_{\mathbb{N}\to\mathbb{A}^*}$ y $Q_{\mathbb{A}^*\to\mathbb{N}}$. Esas cuatro funciones codificadoras nos permiten expresar números naturales como cadenas de símbolos del alfabeto $\mathbb{A}$ considerado. También nos permiten expresar cadenas de símbolos de $\mathbb{A}$ como números naturales. Incluso, podemos definir una palabra de $\mathbb{A}^*$ como un par formado por dos palabras de $\mathbb{A}^*$. Es decir, como un elemento de $(\mathbb{A}^*)^{\times 2}$ o, lo que es lo mismo, de $\mathbb{A}^* \times \mathbb{A}^*$. En la tabla 3.5.1, se muestra la definición de las funciones codificadoras $Q_{\mathbb{A}^*\to(\mathbb{A}^*)^{\times 2}}$ y $Q_{(\mathbb{A}^*)^{\times 2}\to\mathbb{A}^*}$. El alfabeto $\mathbb{A}$ puede ser sustituido por el alfabeto concreto que se esté manejando en cada situación, por ejemplo, $\mathbb{H}$.

Elijamos cualquier lenguaje de programación o cualquier formalismo para expresar algoritmos. Sea $\mathbb{H}$ el alfabeto formado por todos los símbolos utilizables en ese lenguaje de programación o formalismo para expresar algoritmos. Todos los programas y algoritmos expresables son cadenas finitas de símbolos de $\mathbb{H}$. Por tanto, son palabras de $\mathbb{H}^*$. Debido a la existencia de la función biyectiva $Q_{\mathbb{H}^*\to\mathbb{N}}$ (especificada en la tabla 3.4.12, página 63), cada programa o algoritmo puede ser expresado mediante un número natural. Lo mismo ocurre con cualquier dato. Puesto que un dato será una cadena de símbolos de $\mathbb{H}$, el dato es una palabra de $\mathbb{H}^*$ y, por tanto, se puede codificar mediante un número de $\mathbb{N}$ por medio de $Q_{\mathbb{H}^*\to\mathbb{N}}$.

La conclusión final es que, todo programa, algoritmo o dato, es decir, toda la información, puede ser codificada o expresada mediante números naturales.

Especificación de la función biyectiva que asocia a cada palabra de $\mathbb{A}^*$ un par de $(\mathbb{A}^*)^{\times 2}$ teniendo en cuenta las funciones codificadoras $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}$, $\mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}$ y $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}$:
$\mathcal{Q}_{\mathbb{A}^*\to(\mathbb{A}^*)^{\times 2}} : \mathbb{A}^* \to (\mathbb{A}^*)^{\times 2}$ $\mathcal{Q}_{\mathbb{A}^*\to(\mathbb{A}^*)^{\times 2}}(w) = (\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}(x), \mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}(y))$ donde $(x, y) = \mathcal{Q}_{\mathbb{N}\to\mathbb{N}^{\times 2}}(\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}(w))$
Especificación de la función biyectiva que asocia a cada palabra de $(\mathbb{A}^*)^{\times 2}$ un par de $\mathbb{A}^*$ teniendo en cuenta las funciones codificadoras $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}$, $\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}$ y $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}$:
$\mathcal{Q}_{(\mathbb{A}^*)^{\times 2}\to\mathbb{A}^*} : (\mathbb{A}^*)^{\times 2} \to \mathbb{A}^*$ $\mathcal{Q}_{(\mathbb{A}^*)^{\times 2}\to\mathbb{A}^*}(u, v) = \mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}(\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}((\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}(u), \mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}(v))))$
Las especificaciones de las funciones codificadoras $\mathcal{Q}_{\mathbb{A}^*\to\mathbb{N}}$, $\mathcal{Q}_{\mathbb{N}^{\times 2}\to\mathbb{N}}$ y $\mathcal{Q}_{\mathbb{N}\to\mathbb{A}^*}$ se encuentran en las tablas 3.4.12, 3.4.9 y 3.4.11 (páginas 63, 60 y 62).

Tabla 3.5.1. Relación uno-a-uno entre palabras de $\mathbb{A}^*$ y pares de $(\mathbb{A}^*)^{\times 2}$.

3.5.2 Codificación de programas y datos utilizando solo 0 y 1

En el apartado anterior. hemos llegado ala conclusión de que todo programa, todo algoritmo y todo dato puede ser codificado o representado mediante los números naturales. Pero además, como los números naturales pueden ser expresados en sistema binario (utilizando solo 0 y 1), toda la información puede ser codificada o expresada mediante números naturales representados en binario. Consecuentemente, toda la información es expresable mediante 0 y 1.

3.6.

Funciones características e incomputabilidad

3.6.1 Existencia de funciones características no computables

En el apartado 3.3.5, se ha presentado la noción de función característica. En la tabla 3.6.1, se muestra la especificación de la función característica correspondiente a un lenguaje L definido sobre un alfabeto $\mathbb{A}$.

$$\chi_L : \mathbb{A}^* \to 2$$

$$\chi_L(w) = \begin{cases} 1 & \text{si } w \in L \\ 0 & \text{si } w \notin L \end{cases}$$

Tabla 3.6.1. Función característica χ_L para el lenguaje L definido sobre el alfabeto $\mathbb{A}$.

Recordemos que la función característica χ_L devuelve 1 para las palabras pertenecientes al lenguaje L y devuelve 0 para las palabras no pertenecientes al lenguaje L. Formalmente, $2^{\mathbb{A}^*}$ es el conjunto formado por todas las funciones características correspondientes a los lenguajes definidos sobre el alfabeto $\mathbb{A}$.

Para que una función sea considerada computable, se ha de probar la existencia de un algoritmo o programa que realice el cálculo formalizado mediante la especificación de la función.

El razonamiento a seguir par probar la existencia de funciones características de $2^{\mathbb{A}^*}$ que no son computables es el siguiente:

- Para que una función característica perteneciente a $2^{\mathbb{A}^*}$ sea computable, tiene que existir un programa que la compute.

- Para escribir un programa, hay que elegir un lenguaje de programación (Ada, Haskell, Java, etc.). Una vez elegido un lenguaje de programación, el alfabeto o conjunto de símbolos utilizables queda fijado. Supongamos que el conjunto de símbolos utilizables en el lenguaje de programación elegido es $\mathbb{H}$.
- Un programa será una palabra perteneciente a $\mathbb{H}^*$.
- Si una función característica $\chi_L : \mathbb{A}^* \to \mathbb{2}$ perteneciente a $\mathbb{2}^{\mathbb{A}^*}$ es implementable (calculable, computable), entonces existe en $\mathbb{H}^*$ una palabra w_L que es justo un programa (o un algoritmo) que implementa χ_L. Si en $\mathbb{H}^*$ no existe ninguna palabra w_L que sea justo un programa que implemente χ_L, entonces χ_L no es implementable (calculable, computable).
- Puesto que $\mathbb{H}^*$ es enumerable (apartado 3.4.6, página 61) y $\mathbb{2}^{\mathbb{A}^*}$ no es enumerable (apartado 3.4.7, página 66), sabemos que hay menos palabras (programas) en $\mathbb{H}^*$ que funciones características en $\mathbb{2}^{\mathbb{A}^*}$.
- Por tanto, para algunas funciones características de $\mathbb{2}^{\mathbb{A}^*}$ no habrá un programa (una palabra de $\mathbb{H}^*$) que las calcule.
- Esas funciones características de $\mathbb{2}^{\mathbb{A}^*}$ para las que no hay un programa (una palabra de $\mathbb{H}^*$) que las calcule, serán funciones características no computables (incomputables).

La conclusión es que hay funciones características pertenecientes a $\mathbb{2}^{\mathbb{A}^*}$ que no son computables. En la figura 3.6.1 de la página 75, se muestra esta conclusión de manera gráfica.

3.6.2 Funciones características no computables: dos ejemplos

Una vez que se ha probado que existen funciones características no computables, una de las siguientes tareas es encontrar lenguajes —es decir, problemas de decisión— cuyas funciones características son incomputables.

En ese apartado se mencionan dos ejemplos concretos:

1. El problema de decidir si un polinomio general tiene raíces enteras.
2. El problema de decidir si una fórmula de la lógica de primer orden es $False$.

3.6.2.1 El problema de decidir si un polinomio general tiene raíces enteras

En los polinomios generales, los coeficientes serán números enteros y habrá más de una variable. A continuación, se muestra un polinomio general a modo de ejemplo:

$$\delta_1 \equiv 9xy^2 - z^5 + yz^3 + 32$$

$2^{\mathbb{A}^*}$

Conjunto formado por las funciones características

de todos los lenguajes definibles sobre el alfabeto $\mathbb{A}$

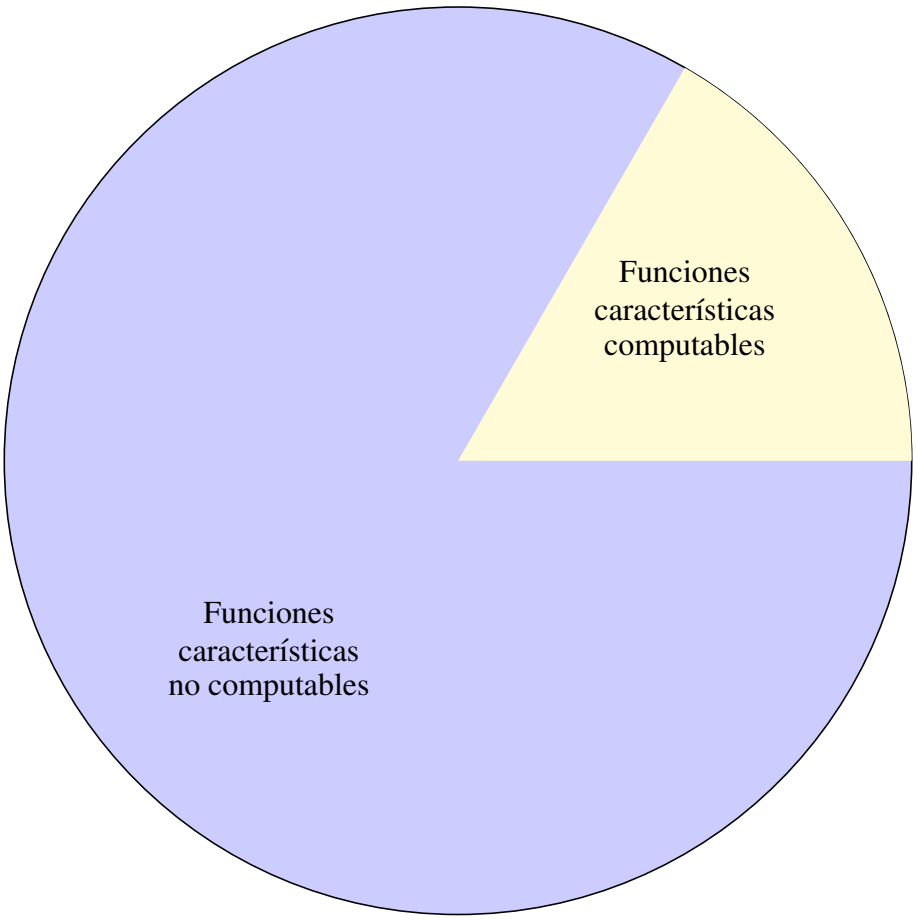


Figura 3.6.1. Funciones características computables y no computables.

Las raíces enteras de ese polinomio son los tripletes [1] ordenados pertenecientes al conjunto $\mathbb{Z}^{\times 3}$ y que representan valores de x, y y z respectivamente. Para ese polinomio, el triplete $(\underbrace{8}_{x}, \underbrace{0}_{y}, \underbrace{2}_{z})$ es una raíz entera. Pero ese polinomio general tiene infinitas raíces enteras. Por ejemplo, todos los tripletes de la forma $(n, 0, 2)$ donde $n \in \mathbb{Z}$ son raíces enteras de ese polinomio general.

Consideremos el siguiente polinomio general:

$$\delta_2 \equiv 3x^2 - 2xy - y^2z - 7$$

El triplete ordenado $(1, 2, -2)$ es una raíz entera para ese polinomio δ_2.

Dado un polinomio general, puede ocurrir que tenga un número finito de raíces o que tenga un número infinito de raíces o que no tenga ninguna raíz.

Por ejemplo, el polinomio general $\delta_3 \equiv x^2 + y^2 + 5z^2 + 3w^2 + 1$ en el que aparecen cuatro variables distintas, no tiene ninguna raíz.

Dado un polinomio general π, la pregunta es la siguiente: ¿Tiene el polinomio general π alguna raíz?. Lo que se querría es obtener la respuesta "Sí, el polinomio general π tiene al menos una raíz" o la respuesta "No, el polinomio general π no tiene ninguna raíz".

Se ha probado matemáticamente que no existe ningún algoritmo general que sea capaz de responder a esa pregunta para todos los polinomios generales. La estrategia más general para buscar raíces de polinomios generales consiste en ir probando con todos los valores posibles de las variables, pero esa estrategia tiene inconvenientes.

Por ejemplo, para un polinomio general en el que aparecen las variables x, y y z, la estrategia sería ir probando con todos los tripletes posibles. Con el propósito de simplificar la explicación, supongamos que se quiere decidir si el polinomio en cuestión tiene alguna raíz cuyos componentes sean no negativos. En ese caso, habría que ir generando los siguientes tripletes: $(0, 0, 0)$, $(0, 0, 1)$, $(0, 1, 0)$, $(1, 0, 0)$, $(0, 0, 2)$, $(0, 1, 1)$, $(0, 2, 0)$, $(1, 0, 1)$, $(1, 1, 0)$, $(2, 0, 0)$, $(0, 0, 3)$, ... Tras generar un triplete, se comprobará si es raíz del polinomio; si es una raíz del polinomio, entonces ya sabemos que el polinomio tiene al menos una raíz y la búsqueda termina y se devuelve la respuesta "Sí"; si el triplete que tenemos entre manos no es raíz del polinomio, entonces se ha de generar el siguiente triplete. La única opción para terminar con el proceso es encontrar un triplete que sea raíz del polinomio. Mientras no se encuentre un triplete que sea raíz del polinomio, habrá que seguir con la búsqueda. Consecuentemente, en el caso de los polinomios generales que no tengan ninguna raíz, la búsqueda será infinita y nunca se sabrá que realmente no hay ninguna raíz. Nunca se podrá terminar el proceso dando la respuesta "No".

[1] Sinónimos de triplete: terna, trío

En resumen, cuando la respuesta ha de ser afirmativa, se conseguirá responder "Sí" tras un número finito de pasos. Pero cuando la respuesta ha de ser negativa, no se conseguirá responder "No" tras un número finito de pasos. En vez de responder "No", el proceso sigue de manera infinita y el usuario que está esperando una respuesta tendrá que seguir esperando sin fin.

El problema de decidir si un polinomio general tiene alguna raíz, se puede formular mediante un lenguaje. Para ello, primero hay que fijar o establecer un alfabeto $\mathbb{H}$ que recoja los símbolos aceptables. Por ejemplo, $\mathbb{H}$ podría ser definido de la siguiente forma:

$$\mathbb{H} = \{-, +, *, 0, 1, 2, \ldots, 9, a, b, c, \ldots, x, y, z\}$$

Ese alfabeto esta formado por operadores matemáticos, dígitos y letras que se utilizarán para dar un nombre a las variables.

Por otro lado, al referirse a un polinomio concreto, conviene diferenciar entre el polinomio como ente matemático y la secuencia de símbolos de $\mathbb{H}$ que se utiliza para representar el polinomio. Consideremos el polinomio $\delta_2 \equiv 3x^2 - 2xy - y^2z - 7$. Ese polinomio entendido como ente matemático lo denotaremos como $\delta_2 \equiv 3x^2 - 2xy - y^2z - 7$. En cambio, para referirnos a la secuencia de símbolos que representan ese polinomio, escribiremos $\langle\delta_2\rangle$:

$$\langle\delta_2\rangle \equiv 3 * x * x - 2 * x * y - y * y * z - 7$$

Por tanto, hay que diferenciar entre δ_2 y $\langle\delta_2\rangle$: δ_2 es el polinomio entendido como ente matemático abstracto; $\langle\delta_2\rangle$ es una palabra de $\mathbb{H}^*$ y representa al polinomio matemático δ_2.

De la misma forma, diferenciaríamos entre el polinomio δ_3 y su representación $\langle\delta_3\rangle$:

$$\delta_3 \equiv x^2 + y^2 + 5z^2 + 3w^2 + 1$$

$$\langle\delta_3\rangle \equiv x * x + y * y + 5 * z * z + 3 * w * w + 1$$

La secuencia de símbolos a la que hemos denominado $\langle\delta_3\rangle$ es una palabra perteneciente a $\mathbb{H}^*$, es decir, $x * x + y * y + 5 * z * z + 3 * w * w + 1$ es una palabra perteneciente a $\mathbb{H}^*$.

Una vez establecido el alfabeto $\mathbb{H}$ y una vez que para cualquier polinomio general π se ha aclarado la diferencia entre π y $\langle\pi\rangle$, el lenguaje L_{pgr} formado por los polinomios generales que tienen al menos una raíz, se definiría de la siguiente forma:

$$L_{pgr} = \{\langle\pi\rangle \mid \langle\pi\rangle \in \mathbb{H}^* \ \wedge \ \pi \text{ es un polinomio general que tiene al menos una raíz}\}$$

Por tanto, cuando nos den una palabra $\langle\pi\rangle$ que representa un polinomio general π, nuestra tarea será decidir si la palabra $\langle\pi\rangle$ pertenece al lenguaje L_{pgr}. Por ejemplo, la palabra $\langle\delta_1\rangle$ y la palabra $\langle\delta_2\rangle$ pertenecen a L_{pgr}, pero la palabra $\langle\delta_3\rangle$ no pertenece al lenguaje L_{pgr}.

La función característica correspondiente sería la siguiente:

$$\chi_{L_{pgr}} : \mathbb{H}^* \to \mathbb{2}$$

$$\chi_{L_{pgr}}(\langle \pi \rangle) = \begin{cases} 1 & \text{si } \pi \text{ es un polinomio general que tiene al menos una raíz} \\ 0 & \text{en caso contrario} \end{cases}$$

En concreto, se cumple $\chi_{L_{pgr}}(\langle \delta_1 \rangle) = 1$, se cumple $\chi_{L_{pgr}}(\langle \delta_2 \rangle) = 1$ y se cumple $\chi_{L_{pgr}}(\langle \delta_3 \rangle) = 0$.

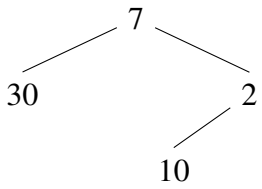
Para insistir en la diferencia entre entes matemáticos abstractos y las secuencias de símbolos que se utilizan para representar esos entes matemáticos abstractos, vamos a recordar el caso de los árboles binarios. Primero, fijamos un alfabeto al que denominaremos $\mathbb{G}$:

$$\mathbb{G} = \{\underbrace{(,)}, \underbrace{\ ,\ }, \underbrace{0,1,2,\ldots,9}, \underbrace{a,b,c,\ldots,x,y,z,A,B,C,\ldots,X,Y,Z}\}$$

Ese alfabeto está formado por los dos símbolos correspondientes a los paréntesis, el símbolo correspondiente a la coma, los dígitos y las letras.

Consideremos el siguiente árbol binario β:

Árbol binario β



La palabra $\langle \beta \rangle$ —perteneciente al conjunto $\mathbb{G}^*$— que utilizaríamos para representar el árbol binario β sería la siguiente:

$$\langle \beta \rangle \equiv Crear(7, Crear(30, Avacio, Avacio), Crear(2, Crear(10, Avacio, Avacio), Avacio)$$

Recordemos que mediante $Avacio$ se representa un árbol binario vacío y que mediante $Crear$ se representa la creación de un árbol binario nuevo a partir de un elemento r que hace de raíz y dos árboles binarios a y b que hacen de subárboles (subárbol izquierdo y subárbol derecho). En la figura 3.6.2 de la página 82 se muestra el diagrama del árbol binario $Crear(r, a, b)$.

El caso del árbol β es un buen ejemplo para entender la diferencia entre el ente matemático abstracto β y la secuencia de símbolos $\langle \beta \rangle$ que se utiliza para representar a β.

3.6.2.2 El problema de decidir si una fórmula de la lógica de primer orden es $False$

El lógica de primer orden se da también el problema de que al intentar diseñar una estrategia para decidir si una fórmula es $False$, en el caso de que la respuesta sea afirmativa se podrá terminar el proceso de decisión respondiendo "Sí" en tiempo finito, pero en el caso de que la respuesta sea negativa no siempre se podrá terminar el proceso de decisión respondiendo "No" en tiempo finito.

Supongamos que tenemos una fórmula de la lógica de primer orden γ y que queremos responder a la siguiente pregunta: ¿Es $False$ el valor de γ?. Querríamos recibir la respuesta "Sí, la fórmula γ es $False$" o, alternativamente, la respuesta "No, la fórmula γ no es $False$".

Se ha probado matemáticamente que no es posible diseñar un algoritmo que sea capaz de averiguar la respuesta correcta (afirmativa o negativa) para todas las fórmulas posibles. La estartegia general que sirve para intentar averiguar si el valor de γ es $False$, consiste en intentar probar que no es posible hacer que γ sea $True$. Pero hay fórmulas para las cuales ese método no sirve para tomar la decisión requerida.

Si no hay manera de hacer que la fórmula γ sea $True$, se podrá llegar a esa conclusión en un número finito de asos y se podrá responder afirmativamente: "Sí, la fórmula γ es $False$". Pero si existe alguna manera de hacer que la fórmula γ sea $True$, entonces puede ocurrir que el proceso para llegar a esa conclusión sea infinito y que, consecuentemente, no se pueda dar la respuesta negativa que querríamos que se diese: "No, la fórmula γ no es $False$".

En resumen, cuando la respuesta ha de ser afirmativa, se conseguirá responder "Sí, la fórmula γ es $False$" tras un número finito de pasos. Pero cuando la respuesta ha de ser negativa, no siempre se conseguirá responder "No, la fórmula γ no es $False$" tras un número finito de pasos. En ese caso, en vez de responder "No", el proceso sigue de manera infinita y el usuario que está esperando una respuesta tendrá que seguir esperando sin fin.

A continuación se muestran tres fórmulas que tienen esa problemática:

- $\rho_1 \equiv \forall x(P(x) \rightarrow \exists y(Q(y) \ \wedge \ R(x,y)))$
- $\rho_2 \equiv \forall x(S_1(x) \rightarrow \exists y(S_2(y) \ \wedge \ (S_3(x,y) \ \vee \ S_4(x,y))))$
- $\rho_3 \equiv (\exists x(P(x))) \ \wedge \ (\forall y(P(y) \rightarrow P(f(y))))$

El problema de decidir si una fórmula de la lógica de primer orden es $False$, se puede formular mediante un lenguaje. Para ello, primero hay que fijar o establecer un alfabeto $\mathbb{J}$ que recoja los símbolos aceptables. Por ejemplo, $\mathbb{J}$ podría ser definido de la siguiente forma:

$$\mathbb{J} = \{\underbrace{\neg, \wedge, \vee, \rightarrow}, \underbrace{\forall, \exists}, \underbrace{(,)}, \underbrace{\ ,\ }, \underbrace{\_}, \underbrace{0,1,2,\ldots,9}, \underbrace{a,b,c,\ldots,x,y,z,A,B,C,\ldots,X,Y,Z}\}$$

Ese alfabeto está formado por los dos símbolos correspondientes a los paréntesis, la coma y el guion bajo y por operadores lógicos, dígitos y letras.

También al referirse a una fórmula lógica concreta, hay que diferenciar entre la fórmula lógica como ente matemático y la secuencia de símbolos de $\mathbb{J}$ que se utiliza para representar la fórmula lógica. Consideremos la fórmula lógica $\rho_1 \equiv \forall x(P(x) \rightarrow \exists y(Q(y) \ \wedge \ R(x,y)))$. Esa fórmula lógica entendida como ente matemático la denotaremos como $\rho_1 \equiv \forall x(P(x) \rightarrow \exists y(Q(y) \ \wedge \ R(x,y)))$. En cambio, para referirnos a la secuencia de símbolos que representan esa fórmula lógica, escribiremos $\langle \rho_1 \rangle$:

$$\langle \rho_1 \rangle \equiv \forall x(P(x) \rightarrow \exists y(Q(y) \ \wedge \ R(x,y)))$$

En este ejemplo concreto, ρ_1 y $\langle \rho_1 \rangle$ son iguales. De todas formas, entenderemos que ρ_1 es el ente matemático abstracto y que $\langle \rho_1 \rangle$ es una secuencia de símbolos —una palabra de $\mathbb{J}^*$— que es una representación del ente matemático ρ_1.

En el caso de la fórmula ρ_2, su representación $\langle \rho_2 \rangle$ será distinta porque asumimos que los subíndices no pueden ser utilizados y que en su lugar hay que poner el guion bajo y el número corresponpiente:

$$\rho_2 \equiv \forall x(S_1(x) \rightarrow \exists y(S_2(y) \ \wedge \ (S_3(x,y) \ \vee \ S_4(x,y))))$$

$$\langle \rho_2 \rangle \equiv \forall x(S\_1(x) \rightarrow \exists y(S\_2(y) \ \wedge \ (S\_3(x,y) \ \vee \ S\_4(x,y))))$$

La secuencia de símbolos $\langle \rho_2 \rangle$ es una palabra del conjunto $\mathbb{J}^*$.

En el caso de la fórmula ρ_3, tenemos que ρ_3 y $\langle \rho_3 \rangle$ son iguales:

$$\rho_3 \equiv (\exists x(P(x))) \ \wedge \ (\forall y(P(y) \rightarrow P(f(y))))$$

$$\langle \rho_3 \rangle \equiv (\exists x(P(x))) \ \wedge \ (\forall y(P(y) \rightarrow P(f(y))))$$

La secuencia de símbolos $\langle \rho_3 \rangle$ es una palabra del conjunto $\mathbb{J}^*$.

Una vez establecido el alfabeto $\mathbb{J}$ y una vez que para cualquier fórmula de la lógica de primer orden γ se ha aclarado la diferencia entre γ y $\langle \gamma \rangle$, el lenguaje L_{fsF} formado por las fórmulas que son False, se definiría de la siguiente forma:

$$L_{fsF} = \{ \langle \gamma \rangle \mid \langle \gamma \rangle \in \mathbb{J}^* \ \wedge \ \text{el valor de la fórmula } \gamma \text{ es } False \}$$

Por tanto, cuando nos den una palabra $\langle \gamma \rangle$ que representa una fórmula lógica γ, nuestra tarea será decidir si la palabra $\langle \gamma \rangle$ pertenece al lenguaje L_{fsF}. Por ejemplo, la palabra $\langle \rho_1 \rangle$, la palabra $\langle \rho_2 \rangle$ y la palabra $\langle \rho_3 \rangle$ no pertenecen a L_{fsF}.

La función característica correspondiente sería la siguiente:

$$\chi_{L_{fsF}} : \mathbb{J}^* \to \mathbb{2}$$

$$\chi_{L_{fsF}}(\langle\gamma\rangle) = \begin{cases} 1 & \text{si el valor de la fórmula } \gamma \text{ es } False \\ 0 & \text{en caso contrario} \end{cases}$$

En concreto, se cumple $\chi_{L_{fsF}}(\langle\rho_1\rangle) = 0$, se cumple $\chi_{L_{fsF}}(\langle\rho_2\rangle) = 0$ y se cumple $\chi_{L_{fsF}}(\langle\rho_3\rangle) = 0$.

3.6.3 Existencia de funciones características indescriptibles

Una función característica es descriptible —o especificable— si existe una espresión, es decir, una cadena de símbolos que la describe.

El razonamiento a seguir par probar la existencia de funciones características de $\mathbb{2}^{\mathbb{A}^*}$ que no son descriptibles es el siguiente:

- Para que una función característica perteneciente a $\mathbb{2}^{\mathbb{A}^*}$ sea descriptible, tiene que existir un expresión —secuencia de símbolos— que la describa.
- Para escribir una expresión, hay que elegir un lenguaje formal (por ejemplo, el lenguaje de la lógica de primer orden) o, si se prefiere, un lenguaje informal (por ejemplo, el castellano). Una vez elegido un lenguaje formal o un lenguaje informal, el alfabeto o conjunto de símbolos utilizables queda fijado. Supongamos que el conjunto de símbolos utilizables en el lenguaje formal o el lenguaje informal elegido es $\mathbb{J}$.
- La descripción o especificación correspondiente a una función característica será una palabra perteneciente a $\mathbb{J}^*$.
- Si una función característica $\chi_L : \mathbb{A}^* \to \mathbb{2}$ perteneciente a $\mathbb{2}^{\mathbb{A}^*}$ es descriptible o especificable, entonces existe en $\mathbb{J}^*$ una palabra w_L que describe a la función característica χ_L. Si en $\mathbb{J}^*$ no existe ninguna palabra w_L que describa a la función característica χ_L, entonces χ_L no es descriptible o especificable.
- Puesto que $\mathbb{J}^*$ es enumerable (apartado 3.4.6, página 61) y $\mathbb{2}^{\mathbb{A}^*}$ no es enumerable (apartado 3.4.7, página 66), sabemos que hay menos palabras (descripciones o especificaciones) en $\mathbb{J}^*$ que funciones características en $\mathbb{2}^{\mathbb{A}^*}$.
- Por tanto, para algunas funciones características de $\mathbb{2}^{\mathbb{A}^*}$ no habrá un una expresión (una palabra de $\mathbb{J}^*$) que las describa.
- Esas funciones características de $\mathbb{2}^{\mathbb{A}^*}$ para las que no hay una expresión (una palabra de $\mathbb{J}^*$) que las describa, serán funciones características no descriptibles (indescriptibles).

En la figura 3.6.3 de la página 83, se muestra esta conclusión de manera gráfica.

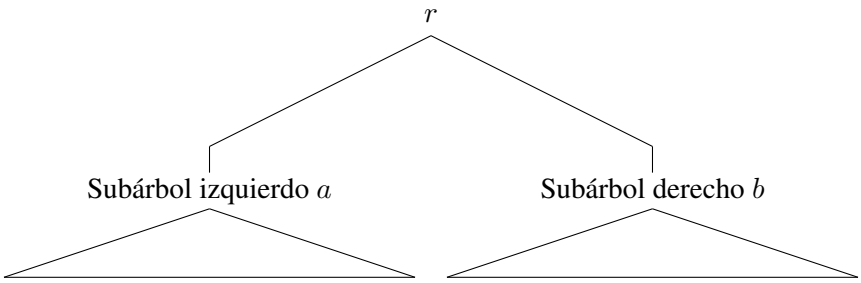


Figura 3.6.2. Diagrama del árbol binario $Crear(r, a, b)$.

$2^{\mathbb{A}^*}$

Conjunto formado por las funciones características

de todos los lenguajes definibles sobre el alfabeto $\mathbb{A}$

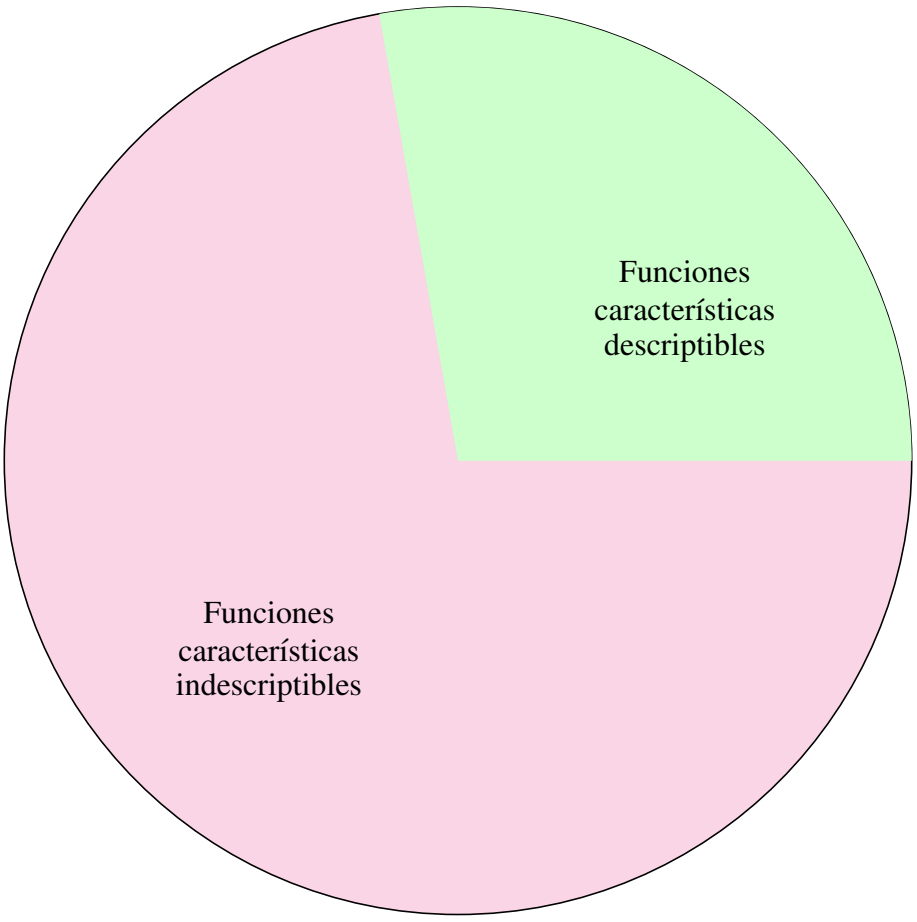


Figura 3.6.3. Funciones características descriptibles e indescriptibles.

3.7.

Funciones semicaracterísticas e incomputabilidad

3.7.1 Funciones semicaracterísticas

En el apartado anterior, se ha expuesto que para el problema de decidir si un polinomio general tiene alguna raíz y para el problema de decidir si una fórmula de lógica de primer orden es $False$, se tiene la correspondiente estrategia que permite responder con un "Sí" siempre que la respuesta ha de ser afirmativa. Pero en algunos casos en los que la respuesta ha de ser negativa, esas estrategias no consiguen terminar con un "No" y se sumergen en un proceso de cálculo infinito que no producirá ninguna respuesta y que nos mantendrá a la espera de manera infinita. Además, se ha probado matemáticamente que no se puede diseñar una estrategia mejor. En otros problemas de decisión ocurre lo mismo.

Ante esa situación, se ha realizado el siguiente planteamiento:

> Tal vez, exigir o esperar que la función característica sea implementable o computable es demasiado. Se han identificado muchos problemas de decisión para los cuales se tienen estrategias que solo funcionan bien en los casos en los que la respuesta ha de ser afirmativa, mientras que no siempre funcionan bien en los casos en los que la respuesta ha de ser negativa. En concreto, en los casos en los que la respuesta ha de ser negativa, a veces el proceso de cálculo es infinito y no genera ninguna respuesta. Por tanto conviene considerar funciones que se adaptan a esa situación en vez de limitarse a utilizar solo funciones características.

A partir de ese planteamiento, surge la noción de función semicaracterística.

Dado un lenguaje L, denotaremos como Σ_L a la función semicaracterística correspondiente al lenguaje L. Recordemos que la definición de un lenguaje L tendrá la siguiente forma:

$$L = \{w \mid w \in \mathbb{A}^* \ \wedge \ P(w)\}$$

En esa definición, P es una propiedad y una palabra w del conjunto $\mathbb{A}^*$ pertenezcerá a L si y solo si w cumple la propiedad P.

Recordemos también que la función característica correspondiente a L se define de la siguiente manera:

$$\chi_L : \mathbb{A}^* \to 2$$

$$\chi_L(w) = \begin{cases} 1 & \text{si se cumple } P(w) \\ 0 & \text{si no se cumple } P(w) \end{cases}$$

Por su parte, la función semicaracterística correspondiente a L se define de la siguiente forma:

$$\Sigma_L : \mathbb{A}^* \to \{1, \perp\}$$

$$\Sigma_L(w) = \begin{cases} 1 & \text{si se cumple } P(w) \\ \perp & \text{si no se cumple } P(w) \end{cases}$$

donde el símbolo $\perp$ representa una computación que no termina nunca y que no genera ninguna respuesta. Por tanto, para las palabras del conjunto $\mathbb{A}^*$ que no pertenezcan a L, la función característica no devolverá ninguna respuesta porque se sumergirá en un proceso infinito de cálculo. Consecuentemente, el valor 1 significa "Sí" —es decir, $True$— y $\perp$ significa que no se generará ninguna respuesta y, además, que el proceso de computación —que será infructuoso—no tendrá fin.

3.7.2 $\{1, \perp\}^{\mathbb{A}^*}$ no es enumerable

Para probar que $2^{\mathbb{A}^*}$ no es enumerable se ha utilizado la técnica de la contradicción. Pero para probar que $\{1, \perp\}^{\mathbb{A}^*}$ no es enumerable vamos a definir una función biyectiva f entre $2^{\mathbb{A}^*}$ y $\{1, \perp\}^{\mathbb{A}^*}$:

$$f : 2^{\mathbb{A}^*} \to \{1, \perp\}^{\mathbb{A}^*}$$

- A cada función característica $\chi : \mathbb{A}^* \to \{0, 1\}$ de $2^{\mathbb{A}^*}$, la función biyectiva f le hace corresponder una función semicaracterística Σ de $\{1, \perp\}^{\mathbb{A}^*}$, estableciendo una relación uno-a-uno:

$$\Sigma : \mathbb{A}^* \to \{1, \perp\}$$

$$\Sigma(w) = \begin{cases} 1 & \text{si } \chi(w) = 1 \\ \perp & \text{si } \chi(w) = 0 \end{cases}$$

- Por existir la función biyectiva f, sabemos que los conjuntos $2^{\mathbb{A}^*}$ y $\{1, \perp\}^{\mathbb{A}^*}$ tienen el mismo número de elementos porque por cada $\chi \in 2^{\mathbb{A}^*}$, hay un $\Sigma \in \{1, \perp\}^{\mathbb{A}^*}$ y viceversa.
- Puesto que $2^{\mathbb{A}^*}$ no es enumerable, $\{1, \perp\}^{\mathbb{A}^*}$ tampoco lo es.

3.7.3 Existencia de funciones semicaracterísticas no computables

- Podemos deducir que existen funciones semicaracterísticas no computables mediante el siguiente razonamiento:
  - Para que una función semicaracterística perteneciente a $\{1, \perp\}^{\mathbb{A}^*}$ sea computable, tiene que existir un programa que la compute.
  - Para escribir un programa, hay que elegir un lenguaje de programación (Ada, Haskell, Java, etc.). Una vez elegido un lenguaje de programación, el alfabeto o conjunto de símbolos utilizables queda fijado. Supongamos que el conjunto de símbolos utilizables en el lenguaje de programación elegido es $\mathbb{H}$.
  - Un programa será una palabra perteneciente a $\mathbb{H}^*$.
  - Puesto que $\mathbb{H}^*$ es enumerable y $\{1, \perp\}^{\mathbb{A}^*}$ no es enumerable, hay menos palabras (programas) en $\mathbb{H}^*$ que funciones semicaracterísticas en $\{1, \perp\}^{\mathbb{A}^*}$.
  - Por tanto, para algunas funciones semicaracterísticas no habrá un programa (una palabra de $\mathbb{H}^*$) que las calcule.
  - Esas funciones semicaracterísticas para las que no hay un programa (una palabra de $\mathbb{H}^*$) que las calcule, serán funciones semicaracterísticas no computables (incomputables).
- En la figura 3.7.1, se muestra esta conclusión de manera gráfica.

3.7.4 Existencia de funciones semicaracterísticas indescriptibles

Una función semicaracterística es descriptible —o especificable— si existe una espresión, es decir, una cadena de símbolos que la describe.

Para probar que en el conjunto $\{1, \perp\}^{\mathbb{A}^*}$ existen funciones semicaracterísticas indescriptibles o no especificables, una opción es adaptar la demostración del apartado 3.6.3 de la página 81 que sirve para probar que existen funciones características indescriptibles:

- Para que una función semicaracterística perteneciente a $\{1, \perp\}^{\mathbb{A}^*}$ sea descriptible, tiene que existir un expresión —secuencia de símbolos— que la describa.
- Para escribir una expresión, hay que elegir un lenguaje formal (por ejemplo, el lenguaje de la lógica de primer orden) o, si se prefiere, un lenguaje informal (por ejemplo, el castellano). Una vez elegido un lenguaje formal o un lenguaje informal, el alfabeto o conjunto de símbolos utilizables queda fijado. Supongamos que el conjunto de símbolos utilizables en el lenguaje formal o el lenguaje informal elegido es $\mathbb{J}$.

$$\{1, \perp\}^{\mathbb{A}^*}$$

Conjunto formado por las funciones semicaracterísticas

de todos los lenguajes definibles sobre el alfabeto $\mathbb{A}$

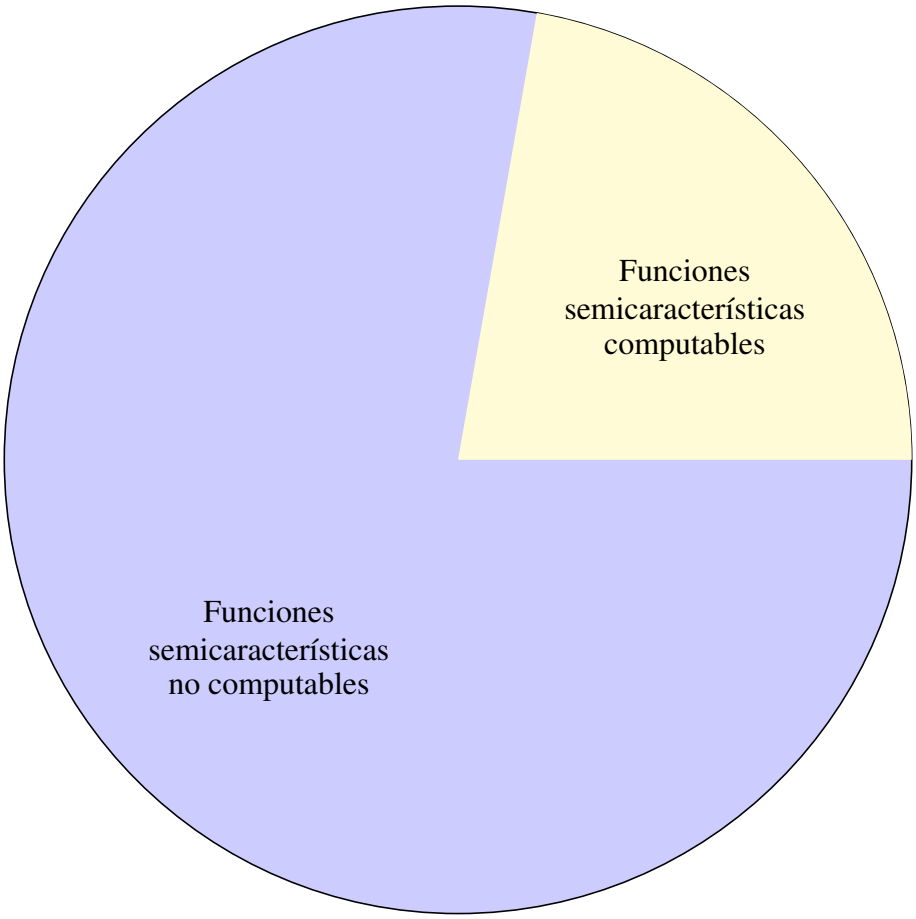


Figura 3.7.1. Funciones semicaracterísticas computables y no computables.

- La descripción o especificación correspondiente a una función semicaracterística será una palabra perteneciente a $\mathbb{J}^*$.
- Si una función semicaracterística $\Sigma_L : \mathbb{A}^* \to \{1, \perp\}$ perteneciente a $\{1, \perp\}^{\mathbb{A}^*}$ es descriptible o especificable, entonces existe en $\mathbb{J}^*$ una palabra w_L que describe a la función semicaracterística χ_L. Si en $\mathbb{J}^*$ no existe ninguna palabra w_L que describa a la función semicaracterística Σ_L, entonces Σ_L no es descriptible o especificable.
- Puesto que $\mathbb{J}^*$ es enumerable (apartado 3.4.6, página 61) y $\{1, \perp\}^{\mathbb{A}^*}$ no es enumerable (apartado 3.7.3, página 87), sabemos que hay menos palabras (descripciones o especificaciones) en $\mathbb{J}^*$ que funciones semicaracterísticas en $\{1, \perp\}^{\mathbb{A}^*}$.
- Por tanto, para algunas funciones semicaracterísticas de $\{1, \perp\}^{\mathbb{A}^*}$ no habrá un una expresión (una palabra de $\mathbb{J}^*$) que las describa.
- Esas funciones semicaracterísticas de $\{1, \perp\}^{\mathbb{A}^*}$ para las que no hay una expresión (una palabra de $\mathbb{J}^*$) que las describa, serán funciones semicaracterísticas no descriptibles (indescriptibles).

En la figura 3.7.2 de la página 90, se muestra esta conclusión de manera gráfica.

3.7.5 Relación entre las propiedades de las funciones características y semicaracterísticas

3.7.5.1 La cantidad de funciones características y semicaracterísticas coincide

Tal como se ha indicado en el apartado 87 de la página 3.7.3, existe una función biyectiva $f : 2^{\mathbb{A}^*} \to \{1, \perp\}^{\mathbb{A}^*}$ que va del conjunto $2^{\mathbb{A}^*}$ al conjunto $\{1, \perp\}^{\mathbb{A}^*}$.

A cada función característica χ_L del conjunto $2^{\mathbb{A}^*}$, la función biyectiva f le asigna una función semicaracterística Σ_L del conjunto $\{1, \perp\}^{\mathbb{A}^*}$. El comportamiento de la función característica χ_L sería el siguiente:

$$\chi_L : \mathbb{A}^* \to \{0, 1\}$$

$$\chi_L(w) = \begin{cases} 1 & \text{si } w \in L \\ 0 & \text{si } w \notin L \end{cases}$$

La función f asigna a la función característica χ_L una función semicaracterística Σ_L cuyo comportamiento sería el siguiente:

$$\Sigma_L : \mathbb{A}^* \to \{1, \perp\}$$

$$\Sigma_L(w) = \begin{cases} 1 & \text{si } \chi_L(w) = 1 \\ \perp & \text{si } \chi_L(w) = 0 \end{cases}$$

$\{1, \perp\}^{\mathbb{A}^*}$

Conjunto formado por las funciones semicaracterísticas

de todos los lenguajes definibles sobre el alfabeto $\mathbb{A}$

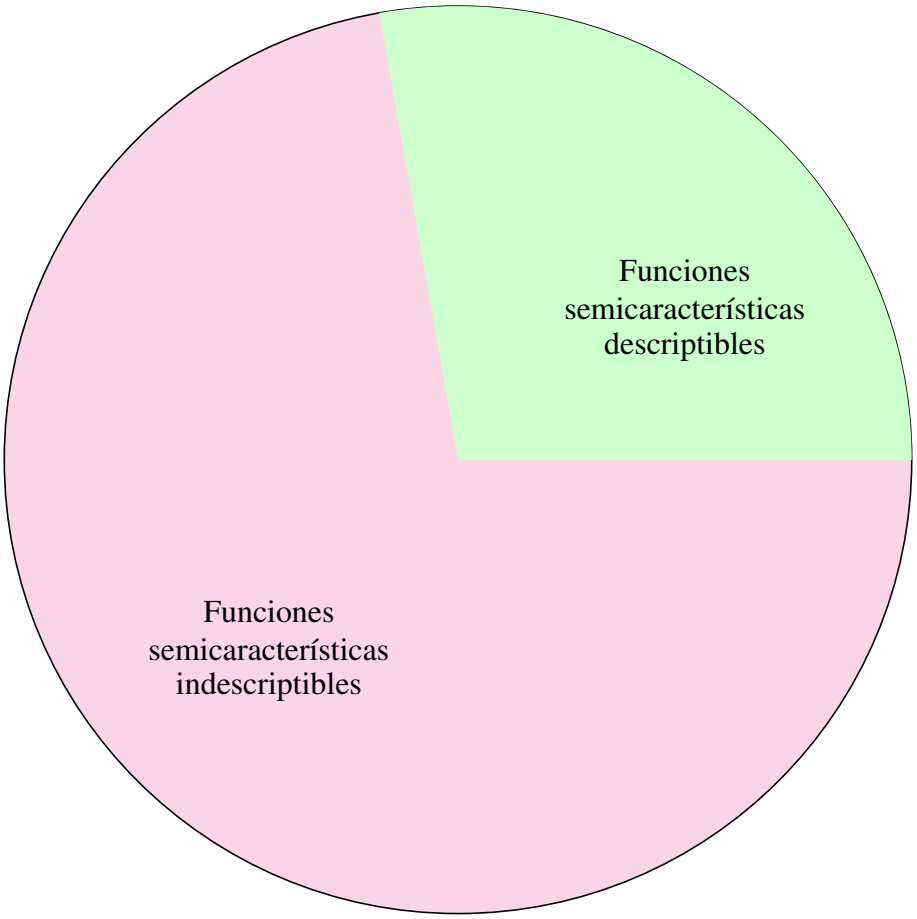


Figura 3.7.2. Funciones semicaracterísticas descriptibles e indescriptibles.

Hay que tener en cuenta que $\Sigma_L = f(\chi_L)$.

Por otro lado, a cada función semicaracterística Σ_L del conjunto $\{1, \perp\}^{\mathbb{A}^*}$, la función biyectiva f^{-1} le asigna una función característica χ_L del conjunto $2^{\mathbb{A}^*}$. El comportamiento de la función semicaracterística Σ_L sería el siguiente:

$$\Sigma_L : \mathbb{A}^* \to \{1, \perp\}$$

$$\Sigma_L(w) = \begin{cases} 1 & \text{si } w \in L \\ \perp & \text{si } w \notin L \end{cases}$$

La función f^{-1} asigna a la función semicaracterística Σ_L una función característica χ_L cuyo comportamiento sería el siguiente:

$$\chi_L : \mathbb{A}^* \to \{0, 1\}$$

$$\chi_L(w) = \begin{cases} 1 & \text{si } \Sigma_L(w) = 1 \\ 0 & \text{si } \Sigma_L(w) = \perp \end{cases}$$

Hay que tener en cuenta que $\chi_L = f^{-1}(\Sigma_L)$.

3.7.5.2 Si una función característica es descriptible, la correspondiente función semicaracterística es descriptible

Si una función característica χ_L es descriptible, entonces la función semicaracterística $\Sigma_L = f(\chi_L)$ será descriptible: para obtener la descripción de la función semicaracterística $\Sigma_L = f(\chi_L)$ es suficiente con poner $\perp$ donde en la descripción de la función característica χ_L se menciona el resultado 0.

3.7.5.3 Si una función semicaracterística es descriptible, la correspondiente función característica es descriptible

Si una función semicaracterística Σ_L es descriptible, entonces la función característica $\chi_L = f^{-1}(\Sigma_L)$ será descriptible: para obtener la descripción de la función característica $\chi_L = f^{-1}(\Sigma_L)$ es suficiente con poner 0 donde en la descripción de la función semicaracterística Σ_L se menciona el resultado $\perp$.

3.7.5.4 Si una función característica es computable, la correspondiente función semicaracterística es computable

Si una función característica χ_L es computable, entonces la función semicaracterística $\Sigma_L = f(\chi_L)$ será computable: para obtener el programa que computa la función semicaracterística $\Sigma_L = f(\chi_L)$ es suficiente con generar un bucle infinito donde en el programa que computa la función característica χ_L se obtiene el resultado 0.

3.7.5.5 El que una función semicaracterística sea computable, no garantiza que la correspondiente función característica sea computable

Aunque una función semicaracterística Σ_L sea computable, no se puede asegurar, en general, que la función característica $\chi_L = f^{-1}(\Sigma_L)$ sea computable. Sabemos que el programa que computa una función semicaracterística Σ_L no devuelve ninguna respuesta —cicla de manera infinita— para algunos datos de entrada, pero, en general, no podemos saber para qué datos se van a poner en marcha esos bucles infinitos. Por tanto, no tenemos manera de sustituir esos bucles infinitos por la respuesta 0. Consecuentemente, con la teoría que tenemos en este tema, no podemos probar que a partir de un programa que compute una función semicaracterística Σ_L se pueda diseñar un programa que compute una función característica $\chi_L = f^{-1}(\Sigma_L)$. Pero tampoco tenemos las herramientas necesarias para probar que existen funciones semicaracterísticas computables cuyas funciones características correspondientes son incomputables.

3.7.6 Relación entre computabilidad y descriptibilidad en las funciones características

3.7.6.1 Si una función característica es computable, entonces es descriptible

Si una función característica es computable, entonces existe un programa que la computa o calcula. En general, a partir de ese programa será posible inferir una descripción de lo que hace el programa. Pero si no es posible inferior una descripción, el propio programa serviría como descripción.

3.7.6.2 El que una función característica sea descriptible no garantiza que sea computable

Aunque se tenga la descripción de qué hace una función característica, no podemos asegurar que se pueda diseñar un algoritmo o programa que realice el cáculo descrito.

3.7.7 Relación entre computabilidad y descriptibilidad en las funciones semicaracterísticas

3.7.7.1 Si una función semicaracterística es computable, entonces es descriptible

Si una función semicaracterística es computable, entonces existe un programa que la computa o calcula. En general, a partir de ese programa será posible inferir una descripción de lo que hace el programa. Pero si no es posible inferior una descripción, el propio programa serviría como descripción.

3.7.7.2 El que una función semicaracterística sea descriptible no garantiza que sea computable

Aunque se tenga la descripción de qué hace una función semicaracterística, no podemos asegurar que se pueda diseñar un algoritmo o programa que realice el cáculo descrito.

3.7.8 La necesidad de utilizar modelos de computación

Tal como se ha mostrado en los apartados anteriores, existe una relación directa entre las funciones características descriptibles y las funciones semicaracterísticas descriptibles. Pero cuando nos trasladamos al área de la computabilidad, no es posible garantizar esa relación directa.

Además, dentro de las funciones características, no se puede garantizar que la descriptibilidad y la computabilidad coincidan. Ocurre lo mismo para las funciones semicaracterísticas.

Al final de este tema, tenemos planteadas las siguientes dos preguntas:

> (1) ¿En el caso de algunas funciones semicaraterísticas computables, ocurre que la correspondiente función característica es incomputable?. Puesto de otra forma, ¿aunque una función semicaracterística Σ_L sea computable, puede ocurrir que la función característica $\chi_L = f^{-1}(\Sigma_L)$ sea incomputable? Esta pregunta se puede formular también de la siguiente forma: ¿Existen problemas que no son computables pero sí son semicomputables?
>
> (2) ¿Algunas funciones semicaraterísticas descriptibles, son incomputables? Puesto de otra forma, ¿existen problemas que no son ni siquiera semicomputables?

La teoría presentada en este Tema 3 no sirve para responder a esas preguntas. Para poder responder a esas preguntas, es necesario adentrarse en las técnicas de diseño de algoritmos y programas. Con ese objetivo se suelen utilizar los modelos computacionales. Con un modelo computacional evitaremos el tener que elegir un lenguaje de programación concreto. Las características particulares de un lenguaje de programación pueden quedar obsoletas con el tiempo. Además, los lenguajes de programación ofrecen muchos detalles que son innecesarios para estudiar la computabilidad. En cambio, en los modelos de computación solo se tiene lo que es fundamental y estrictamente necesario para estudiar la computabilidad y se prescinde de detalles innecesarios. De esa manera se consigue también que los modelos de computación no queden obsoletos con el tiempo.

3.8.

Resumen

El primer paso en el proceso de automatización de una función o de un cálculo, es formular su descripción, es decir, su especificación. Tras formular la descripción, habrá que diseñar un algoritmo que realice la tarea descrita mediante la especificación.

Las descripciones y los algoritmos son secuencias de símbolos, es decir, palabras. Los conjuntos formados por secuencias de símbolos son lenguajes.

Puesto que un lenguaje es un conjunto, a cada lenguaje le corresponde su función característica. Dada una palabra, la función característica correspondiente a un lenguaje devolverá 1 si la palabra pertenece al lenguaje en cuestión y devolverá 0 si la palabra no pertenece al lenguaje en cuestión.

Un lenguaje puede ser representado de dos formas: como un conjunto o por medio de su función característica.

Cada lenguaje —o si se prefiere, la función característica correspondiente a cada lenguaje— puede ser entendido como un problema de decisión.

De la misma forma, cualquier problema de decisión puede ser entendido como un lenguaje. La especificación de cada problema de decisión puede ser formulado a modo de lenguaje.

Por tanto, en este tema, al estudiar la teoría sobre lenguajes, estamos estudiando la teoría sobre los problemas de decisión.

Por otro lado, a cada lenguaje le corresponde también su función semicaracterística. Dada una palabra, la función semicaracterística correspondiente a un lenguaje devolverá 1 si la palabra pertenece al lenguaje en cuestión y no devolverá ninguna respuesta —se sumergirá en un proceso infinito de cálculo— si la palabra no pertenece al lenguaje en cuestión.

A cada lenguaje le corresponde una única función característica y a cada función característica le corresponde un único lenguaje. Asimismo, a cada lenguaje le corresponde una única

función semicaracterística y a cada función semicaracterística le corresponde un único lenguaje.

El conjunto $\mathbb{A}^*$ de todas las palabras definibles sobre un alfabeto $\mathbb{A}$ es enumerable.

El conjunto $\mathsf{Sub}(\mathbb{A}^*)$ de todos los lenguajes definibles sobre un alfabeto $\mathbb{A}$ —es decir, el conjunto formado por todos los subconjuntos de $\mathbb{A}^*$— no es enumerable. Consecuentemente, el conjunto $2^{\mathbb{A}^*}$ de todas las funciones semicaracterísticas de los lenguajes definibles sobre un alfabeto $\mathbb{A}$ no es enumerable y el conjunto $\{1, \perp\}^{\mathbb{A}^*}$ de todas las funciones semicaracterísticas de los lenguajes definibles sobre un alfabeto $\mathbb{A}$ tampoco es enumerable.

Un lenguaje es decidible si existe un algoritmo para su función característica. Un lenguaje es semidecidible —o reconocible— si existe un algoritmo para su función semicaracterística.

Una función característica es descriptible o especificable si existe una expresión que la describe.

Una expresión que describe una función característica es una secuencia de símbolos, es decir, una palabra.

Como hay más funciones características que palabras, hay funciones características para las cuales no existe ninguna expresión que las describa. Esas funciones características son indescriptibles.

Una función característica es computable si existe un algoritmo que la calcule.

Un programa es una secuencia de símbolos, es decir, una palabra.

Puesto que el número de funciones características es mayor que el de palabras, hay funciones características para las cuales no existe ningún algoritmo capaz de calcularlas. Esas funciones características son incomputables.

Dichas conclusiones se extienden también a las funciones semicaracterísticas.

Una función semicaracterística es descriptible o especificable si existe una expresión que la describe.

Una expresión que describe una función semicaracterística es una secuencia de símbolos, es decir, una palabra.

Como hay más funciones semicaracterísticas que palabras, hay funciones semicaracterísticas para las cuales no existe ninguna expresión que las describa. Esas funciones semicaracterísticas son indescriptibles.

Una función semicaracterística es computable si existe un algoritmo que la calcule.

Un programa es una secuencia de símbolos, es decir, una palabra.

Puesto que el número de funciones semicaracterísticas es mayor que el de palabras, hay funciones semicaracterísticas para las cuales no existe ningún algoritmo capaz de calcularlas. Esas funciones semicaracterísticas son incomputables.

Para finalizar, podemos obtener las siguientes conclusiones:

- Si una función característica es descriptible, entonces la correspondiente función semicaracterística es también descriptible.
- Si una función semicaracterística es descriptible, entonces la correspondiente función característica es también descriptible.
- Si una función característica es computable, entonces la correspondiente función semicaracterística es también computable.

En cambio, aunque una función semicaracterística sea computable no podemos asegurar que la correspondiente función característica sea también computable.

De hecho, la realidad es que para algunas funciones semicaracterísticas computables ocurre que la correspondiente función característica no es computable. Pero no se puede llegar a probar eso con la teoría presentada en este tema. Necesitamos pasar a los modelos de computación, los cuales nos ofrecen más herramientas para analizar casos concretos de funciones semicaracterísticas y funciones características.

Conjuntos que se han utilizado a modo de tipos de datos:	
$\mathbb{2}$:	conjunto $\{0, 1\}$ (valores Booleanos), donde 0 representa el valor *falso* y 1 el valor *cierto*.
$\{1, \perp\}$:	conjunto $\{1, \perp\}$, donde 1 representa el valor *cierto* y $\perp$ representa un cálculo infinito sin respuesta.
$\mathbb{N}$:	números naturales (0, 1, 2, 3, etc.).
$\mathbb{Z}$:	números enteros (negativos, 0 y positivos).
$\mathbb{R}$:	números reales.
$\mathbb{A}$:	alfabeto o conjunto de símbolos. En vez de $\mathbb{A}$ puede ser cualquier otra letra. Por ejemplo, $\mathbb{H}$.
$\mathbb{A}^*$:	conjunto de todas las palabras definibles sobre el alfabeto $\mathbb{A}$. Puede ser cualquier otra letra. Por ejemplo, $\mathbb{H}^*$ es el conjunto de todas las palabras definibles sobre el alfabeto $\mathbb{H}$.
$\mathbb{2}^{\mathbb{A}^*}$:	conjunto de las funciones características de todos los lenguajes definibles sobre el alfabeto $\mathbb{A}$. Puede ser cualquier otra letra. Por ejemplo, $\mathbb{2}^{\mathbb{H}^*}$ es el conjunto de las funciones características de todos los lenguajes definibles sobre el alfabeto $\mathbb{H}$.
$\{1, \perp\}^{\mathbb{A}^*}$:	conjunto de las funciones semicaracterísticas de todos los lenguajes definibles sobre el alfabeto $\mathbb{A}$. Puede ser cualquier otra letra. Por ejemplo, $\{1, \perp\}^{\mathbb{H}^*}$ es el conjunto de las funciones semicaracterísticas de todos los lenguajes definibles sobre el alfabeto $\mathbb{H}$.
$\mathsf{Sub}(\mathbb{A}^*)$:	conjunto formado por todos los subconjuntos de $\mathbb{A}^*$, es decir, conjunto formado por todos los lenguajes definibles sobre el alfabeto $\mathbb{A}$. Puede ser cualquier otra letra. Por ejemplo, $\mathsf{Sub}(\mathbb{H}^*)$ es el conjunto formado por todos los lenguajes definibles sobre el alfabeto $\mathbb{H}$.

El operador $\times$ representa el producto cartesiano y sirve para generar tuplas.
$\mathbb{N} \times \mathbb{N}$, o $\mathbb{N}^{\times 2}$, es el tipo formado por pares de enteros.
$\mathbb{A}^* \times \mathbb{A}^*$, o $(\mathbb{A}^*)^{\times 2}$, es el tipo formado por pares de palabras definidas sobre el alfabeto $\mathbb{A}$.

Tabla 3.8.1. Tipos de datos utilizados.

3.9.

Ejercicios

3.9.1 Comprensión de definiciones formales de lenguajes

Dar algunas palabras que pertenecen y algunas palabras que no pertenecen a los siguientes lenguajes definidos sobre el alfabeto $\mathbb{A} = \{a, b, c\}$:

1. $H_1 = \{w \mid w \in \mathbb{A}^* \wedge \exists x(x \in \{\alpha\varepsilon | \alpha \in \mathbb{A}\}^2 \wedge w = x \cdot x^R \cdot x)\}$
2. $H_2 = \{w \mid w \in \mathbb{A}^* \wedge w \cdot w = w \cdot w \cdot w\}$
3. $H_3 = \{w \mid w \in \mathbb{A}^* \wedge \exists u, v(u \in \mathbb{A}^* \wedge v \in \mathbb{A}^* \wedge u \cdot v \cdot w = w \cdot v \cdot u)\}$
4. $H_4 = \{w \mid w \in \mathbb{A}^* \wedge \exists u(u \in \mathbb{A}^* \wedge w \cdot w \cdot w = u \cdot u)\}$

3.9.2 Definición formal de lenguajes

Utilizar, según convenga, la notación de conjuntos, las operaciones sobre palabras y las operaciones sobre lenguajes para describir de manera formal los siguientes lenguajes definidos sobre el alfabeto $\mathbb{A} = \{a, b, c\}$. La mayoría de estos ejercicios pertenecen a exámenes de años anteriores y en ellos se indica la puntuación correspondiente, lo cual da una idea de la dificultad relativa:

1. L_1 – Lenguaje formado por las palabras $aa\varepsilon$, $bb\varepsilon$ y $ac\varepsilon$.
2. L_2 – Lenguaje formado por las palabras ε, $bbc\varepsilon$ y $acc\varepsilon$.
3. L_3 – Lenguaje formado por las palabras que contienen exactamente cuatro símbolos (palabras de longitud cuatro). Por ejemplo, las palabras $aaaa\varepsilon$, $bcab\varepsilon$ y $cbbb\varepsilon$ pertenecen al lenguaje L_3 mientras que ε, $a\varepsilon$, $bc\varepsilon$ y $bcbcba\varepsilon$ no.
4. L_4 – Lenguaje formado por las palabras que contienen exactamente cuatro símbolos, de los cuales exactamente uno es a. Por ejemplo, las palabras $abcb\varepsilon$, $ccac\varepsilon$ y $cbca\varepsilon$ pertenecen al lenguaje L_4 mientras que ε, $abc\varepsilon$, $bc\varepsilon$, $aabc\varepsilon$ y $cccbb\varepsilon$ no.

5. L_5 – Lenguaje formado por las palabras que no contienen ningún símbolo repetido. Por ejemplo, las palabras ε, $a\varepsilon$, $ac\varepsilon$ y $acb\varepsilon$ pertenecen al lenguaje L_5 mientras que $aa\varepsilon$, $bcac\varepsilon$ y $accaaa\varepsilon$ no.

6. L_6 (0,075 puntos) Dar la definición formal del lenguaje L_6 formado por las palabras que tienen por lo menos dos símbolos distintos. Por ejemplo, las palabras $aab\varepsilon$, $accccabab\varepsilon$ y $cccbc\varepsilon$ pertenecen al lenguaje L_6 mientras que $aaa\varepsilon$, $b\varepsilon$ y ε no.

7. L_7 (0,100 puntos) Dar la definición formal del lenguaje L_7 formado por las palabras que no tienen dos o más símbolos distintos, es decir, cada palabra del lenguaje está formado por cero o más repeticiones del mismo símbolo. Por ejemplo, ε, $bbb\varepsilon$, $aa\varepsilon$ y $cccc\varepsilon$ pertenecen al lenguaje L_7 mientras que $ac\varepsilon$, $baaa\varepsilon$ y $aaccb\varepsilon$ no.

8. L_8 (0,025 puntos) Dar la definición formal del lenguaje L_8 formado por las palabras que tienen longitud par. Por ejemplo, ε, $ab\varepsilon$, $aaaa\varepsilon$ y $cabb\varepsilon$ pertenecen al lenguaje L_8 mientras que $a\varepsilon$, $bab\varepsilon$ y $accaa\varepsilon$ no.

9. L_9 (0,025 puntos) Dar la definición formal del lenguaje L_9 formado por las palabras que tienen longitud impar. Por ejemplo, $a\varepsilon$, $bab\varepsilon$ y $accaa\varepsilon$ pertenecen al lenguaje L_9 mientras que ε, $ab\varepsilon$, $aaaa\varepsilon$ y $cabb\varepsilon$ no.

10. L_{10} (0,100 puntos) Dar la definición formal del lenguaje L_{10} formado por las palabras que no tienen dos o más símbolos distintos y cuya longitud es par. Por ejemplo, ε, $bbbb\varepsilon$, $aa\varepsilon$ y $cccc\varepsilon$ pertenecen al lenguaje L_{10} mientras que $baaa\varepsilon$, $aaa\varepsilon$ y $aaccb\varepsilon$ no.

11. L_{11} (0,025 puntos) Dar la definición formal del lenguaje L_{11} formado por las palabras que empiezan con el símbolo a. Por ejemplo, $a\varepsilon$, $aa\varepsilon$, $abcc\varepsilon$, $abaa\varepsilon$ y $acb\varepsilon$ pertenecen al lenguaje L_{11} mientras que ε, $bc\varepsilon$ y $cbab\varepsilon$ no.

12. L_{12} (0,025 puntos) Dar la definición formal del lenguaje L_{12} formado por las palabras que no empiezan con el símbolo a. Por ejemplo, ε, $bc\varepsilon$ y $cbab\varepsilon$ pertenecen al lenguaje L_{12} mientras que $a\varepsilon$, $aa\varepsilon$, $abcc\varepsilon$, $abaa\varepsilon$ y $acb\varepsilon$ no.

13. L_{13} (0,025 puntos) Dar la definición formal del lenguaje L_{13} formado por las palabras que terminan con el símbolo a. Por ejemplo, $a\varepsilon$, $ccca\varepsilon$, $aaa\varepsilon$ y $abaa\varepsilon$ pertenecen al lenguaje L_{13} mientras que ε, $aab\varepsilon$, $b\varepsilon$ y $ccc\varepsilon$ no.

14. L_{14} (0,025 puntos) Dar la definición formal del lenguaje L_{14} formado por las palabras que no terminan con el símbolo a. Por ejemplo, ε, $c\varepsilon$, $ccb\varepsilon$, $aac\varepsilon$ y $abac\varepsilon$ pertenecen al lenguaje L_{14} mientras que $a\varepsilon$, $aa\varepsilon$, $baa\varepsilon$ y $acbaaa\varepsilon$ no.

15. L_{15} (0,050 puntos) Dar la definición formal del lenguaje L_{15} formado por las palabras que empiezan con el símbolo a y terminan con el símbolo a. Por ejemplo, $a\varepsilon$, $aa\varepsilon$, $abba\varepsilon$ y $acaaabba\varepsilon$ pertenecen al lenguaje L_{15} mientras que ε, $c\varepsilon$, $ab\varepsilon$, $bbc\varepsilon$ y $ccaa\varepsilon$ no.

16. L_{16} (0,050 puntos) Dar la definición formal del lenguaje L_{16} formado por las palabras que no empiezan ni terminan con el símbolo a. Por ejemplo, ε, $b\varepsilon$, $baac\varepsilon$, $ccc\varepsilon$ y $ccaaabac\varepsilon$ pertenecen al lenguaje L_{16} mientras que $a\varepsilon$, $abb\varepsilon$, $abba\varepsilon$ y $caa\varepsilon$ no.

17. L_{17} – Dar la definición formal del lenguaje L_{17} formado por palabras de longitud impar y cuyo símbolo central sea a. Por ejemplo, $a\varepsilon$, $aaa\varepsilon$, $ababc\varepsilon$ y $ccaabba\varepsilon$ pertenecen al lenguaje L_{17} mientras que ε, $b\varepsilon$, $aa\varepsilon$, $abc\varepsilon$ y $abcc\varepsilon$ no.

18. L_{18} – Dar la definición formal del lenguaje L_{18} formado por las palabras que terminan en b. Por ejemplo, $b\varepsilon$, $aab\varepsilon$, $bbb\varepsilon$ y $bacb\varepsilon$ pertenecen al lenguaje L_{18} mientras que ε, $c\varepsilon$, $ba\varepsilon$, $ccc\varepsilon$ y $abbbc\varepsilon$ no.

19. L_{19} – Dar la definición formal del lenguaje L_{19} formado por las palabras que empiezan por a y terminan en b. Por ejemplo, $ab\varepsilon$, $aaacb\varepsilon$ y $abcab\varepsilon$ pertenecen al lenguaje L_{19} mientras que ε, $a\varepsilon$, $ca\varepsilon$, $bca\varepsilon$ y $bbbcb\varepsilon$ no.

20. L_{20} –Dar la definición formal del lenguaje L_{20} formado por las palabras que empiezan por a o terminan en b. Por ejemplo, $a\varepsilon$, $b\varepsilon$, $ab\varepsilon$, $ac\varepsilon$, $cb\varepsilon$, $aaa\varepsilon$, $aacb\varepsilon$ y $ccb\varepsilon$ pertenecen al lenguaje L_{20} mientras que ε, $c\varepsilon$, $cca\varepsilon$ y $baac\varepsilon$ no.

21. L_{21} – Dar la definición formal del lenguaje L_{21} formado por las palabras que empiezan por a pero que no terminan en b. Por ejemplo, $a\varepsilon$, $ac\varepsilon$, $aaa\varepsilon$, $abbc\varepsilon$ y $abbba\varepsilon$ pertenecen al lenguaje L_{21} mientras que ε, $b\varepsilon$, $ab\varepsilon$, $ccb\varepsilon$ y $cacb\varepsilon$ no.

22. L_{22} (0,025 puntos) Dar la definición formal del lenguaje L_{22} formado por las palabras que contienen más a-s que b-s. Por ejemplo, $a\varepsilon$, $acc\varepsilon$, $baac\varepsilon$ y $aaa\varepsilon$ pertenecen al lenguaje L_{22} mientras que ε, $ab\varepsilon$, $bbac\varepsilon$, $bbb\varepsilon$ y $cccc\varepsilon$ no.

23. L_{23} (0,025 puntos) Dar la definición formal del lenguaje L_{23} formado por las palabras que contienen un número par de a-s. Por ejemplo, ε, $b\varepsilon$, $aa\varepsilon$, $baab\varepsilon$, $caba\varepsilon$, $aaaa\varepsilon$ y $ccc\varepsilon$ pertenecen al lenguaje L_{23} mientras que $a\varepsilon$, $bac\varepsilon$, $aaa\varepsilon$ y $ccab\varepsilon$ no.

24. L_{24} (0,025 puntos) Dar la definición formal del lenguaje L_{24} formado por las palabras que no contienen más a-s que b-s. Por ejemplo, ε, $ab\varepsilon$, $ccc\varepsilon$, $bc\varepsilon$ y $bacc\varepsilon$ pertenecen al lenguaje L_{24} mientras que $a\varepsilon$, $aba\varepsilon$, $ca\varepsilon$ y $aaa\varepsilon$ no.

25. L_{25} (0,075 puntos) Dar la definición formal del lenguaje L_{25} formado por las palabras que contienen más a-s que b-s y cuyo número de a-s es par. Por ejemplo, $aa\varepsilon$, $caba\varepsilon$ y $aaaa\varepsilon$ pertenecen al lenguaje L_{25} mientras que ε, $b\varepsilon$, $aaab\varepsilon$, $ccb\varepsilon$ y $acc\varepsilon$ no.

26. L_{26} (0,025 puntos) Dar la definición formal del lenguaje L_{26} formado por las palabras que no contienen ni b-s ni c-s. Por ejemplo, ε, $a\varepsilon$, $aa\varepsilon$ y $aaa\varepsilon$ pertenecen al lenguaje L_{26} mientras que $b\varepsilon$, $abca\varepsilon$, $ccc\varepsilon$ y $abb\varepsilon$ no.

27. L_{27} (0,025 puntos) Dar la definición formal del lenguaje L_{27} formado por las palabras que no contienen ni a-s ni c-s. Por ejemplo, ε, $b\varepsilon$, $bb\varepsilon$ y $bbbb\varepsilon$ pertenecen al lenguaje L_{27} mientras que $c\varepsilon$, $aaa\varepsilon$, $ac\varepsilon$, $bac\varepsilon$ y $bcc\varepsilon$ no.

28. L_{28} – Dar la definición formal del lenguaje L_{28} formado por las palabras que no contienen ninguna c y en las que todas las apariciones de a (si hay alguna a) están seguidas en la izquierda y todas las apariciones de b (si hay alguna b) están seguidas en la derecha. Por

ejemplo, ε, $aab\varepsilon$, $aaabbbb\varepsilon$, $aaa\varepsilon$ y $bb\varepsilon$ pertenecen al lenguaje L_{28} mientras que $caa\varepsilon$, $abcb\varepsilon$, $bbaaa\varepsilon$ y $ccc\varepsilon$ no.

29. L_{29} (0,200 puntos) Dar la definición formal del lenguaje L_{29} formado por las palabras que no contienen ninguna c y donde todas las a-s (si hay a-s) están juntas (en el lado izquierdo o en el lado derecho) y todas las b-s (si hay b-s) están también juntas. Por ejemplo, ε, $aabbb\varepsilon$, $baaa\varepsilon$, $bbb\varepsilon$ y $aaaa\varepsilon$ son palabras de L_{29} mientras que $aabaa\varepsilon$, $aaaccbb\varepsilon$ y $abaaa\varepsilon$ no son palabras de L_{29}.

30. L_{30} (0,100 puntos) Dar la definición formal del lenguaje L_{30} formado por las palabras que no contienen ninguna c, tienen el mismo número de a-s que de b's y donde todas las a-s (si hay a-s) están juntas (en el lado izquierdo) y todas las b-s (si hay b-s) están también juntas (en el lado derecho). Por ejemplo, ε, $ab\varepsilon$, $aabb\varepsilon$ y $aaabbb\varepsilon$ son palabras de L_{30} mientras que $aabbb\varepsilon$, $aaacbb\varepsilon$, $aaa\varepsilon$ y $bbaa\varepsilon$ no son palabras de L_{30}.

31. L_{31} (0,125 puntos) Dar la definición formal del lenguaje L_{31} formado por las palabras donde todas las a-s (si hay a-s) están juntas (en el lado izquierdo), todas las b-s están también juntas (en el centro) y todas las c-s están también juntas (en el lado derecho). Adicionalmente, el número de b's ha de ser mayor que el de a's y el número de c's ha de ser mayor que el de b's. Por ejemplo, $bccc\varepsilon$, $abbccc\varepsilon$, $aabbbccccccc\varepsilon$ y $bbcccc\varepsilon$ son palabras de L_{31} mientras que ε, $aabbb\varepsilon$, $aaacbb\varepsilon$, $aaa\varepsilon$, $ccc\varepsilon$ y $bbaa\varepsilon$ no son palabras de L_{31}.

32. L_{32} (0,025 puntos) Dar la definición formal del lenguaje L_{32} formado por las palabras en las que el número de a-s coincide con la suma del número de b-s y c-s. Por ejemplo, las palabras ε, $aabc\varepsilon$, $acccaa\varepsilon$ y $cabaca\varepsilon$ pertenecen al lenguaje L_{32} mientras que $aaa\varepsilon$, $b\varepsilon$ y $accb\varepsilon$ no.

33. L_{33} (0,100 puntos) Dar la definición formal del lenguaje L_{33} formado por las palabras en las que después de una a siempre hay al menos dos b-s. Por ejemplo, las palabras ε, $bcbbcabb\varepsilon$, $abbbabbabbc\varepsilon$ y $cccc\varepsilon$ pertenecen al lenguaje L_{33} mientras que $baaa\varepsilon$, $ab\varepsilon$ y $aaccb\varepsilon$ no.

34. L_{34} (0,025 puntos) Dar la definición formal del lenguaje L_{34} formado por las palabras que no contienen ni b-s ni c-s y el número de a-s es par. Por ejemplo, las palabras ε, $aaaa\varepsilon$ y $aa\varepsilon$ pertenecen al lenguaje L_{34} mientras que $baaa\varepsilon$, $bb\varepsilon$, $cbbb\varepsilon$, $c\varepsilon$, $aaa\varepsilon$ y $aaccb\varepsilon$ no.

35. L_{35} – Dar la definición formal del lenguaje L_{35} formado por las palabras que cumplen alguna (por lo menos una) de las siguientes dos condiciones:

    - no contienen ninguna b ni ninguna c
    - el número de a's es par.

    Por ejemplo, las palabras ε, $aaa\varepsilon$, $aaaa\varepsilon$, $abca\varepsilon$, $bb\varepsilon$ y $aa\varepsilon$ pertenecen al lenguaje L_{35} mientras que $baaa\varepsilon$, $bab\varepsilon$, $cbbbaaa\varepsilon$, $ca\varepsilon$, $aaca\varepsilon$ y $aaccba\varepsilon$ no.

36. L_{36} (0,025 puntos) Dar la definición formal del lenguaje L_{36} formado por las palabras que contienen por lo menos una a y una c. Por ejemplo, las palabras $ca\varepsilon$, $aabbbbaabc\varepsilon$ y $ccccaa\varepsilon$ pertenecen al lenguaje L_{36} mientras que ε, $baaa\varepsilon$, $bb\varepsilon$, $cbbb\varepsilon$, $c\varepsilon$ y $aaa\varepsilon$ no.

37. L_{37} (0,050 puntos) Dar la definición formal del lenguaje L_{37} formado por las palabras que contienen al menos una aparición de la cadena ac o la cadena ca. Por ejemplo, las palabras $ca\varepsilon$, $acabbbbccaac\varepsilon$, $abaccb\varepsilon$ y $acaccbaac\varepsilon$ pertenecen al lenguaje L_{37} mientras que ε, $cbaaa\varepsilon$, $bba\varepsilon$, $cbbab\varepsilon$, $bbb\varepsilon$, $c\varepsilon$ y $aaa\varepsilon$ no.

38. L_{38} (0,100 puntos) Dar la definición formal del lenguaje L_{38} formado por las palabras en las que a y c nunca aparecen juntos ni como ac ni como ca. Por ejemplo, las palabras ε, $cbaaa\varepsilon$, $bcba\varepsilon$, $cbbb\varepsilon$, $c\varepsilon$ y $aaa\varepsilon$ pertenecen al lenguaje L_{38} mientras que $ca\varepsilon$, $aabbbbaac\varepsilon$ y $ccccaa\varepsilon$ no.

39. L_{39} (0,025 puntos) Dar la definición formal del lenguaje L_{39} formado por las palabras que contienen el mismo número de a-s que de b-s. Por ejemplo, $aabacbcb\varepsilon$, $ccc\varepsilon$, ε, $aaabbb\varepsilon$, $abab\varepsilon$ y $bccaccc\varepsilon$ pertenecen a L_{39} mientras que $b\varepsilon$, $ca\varepsilon$, $aabbbbca\varepsilon$ y $ccccaa\varepsilon$ no.

40. L_{40} (0,025 puntos) Dar la definición formal del lenguaje L_{40} formado por las palabras que empiezan por a, terminan en b y contienen el mismo número de a-s que de b-s. Por ejemplo, $aabacbcb\varepsilon$, $acb\varepsilon$, $aababb\varepsilon$ y $accbbcaab\varepsilon$ pertenecen a L_{40}. En cambio, $abba\varepsilon$ no pertenece a L_{40} porque, aunque el número de a's y b's es el mismo, la palabra no acaba en b.

41. L_{41} (0,025 puntos) Dar la definición formal del lenguaje L_{41} formado por las palabras que contienen la subpalabra $aa\varepsilon$. Por ejemplo, $aaaaa\varepsilon$, $aabacbcb\varepsilon$, $acaaab\varepsilon$, $cbaabaab\varepsilon$ y $accbaaaab\varepsilon$ pertenecen a L_{41} mientras que ε, $b\varepsilon$, $ca\varepsilon$, $abbbca\varepsilon$ y $cccc\varepsilon$ no.

42. L_{42} (0,075 puntos) Dar la definición formal del lenguaje L_{42} formado por las palabras que contienen la subpalabra $aa\varepsilon$ y la subpalabra $cc\varepsilon$ (pero puede aparecer antes $cc\varepsilon$ que $aa\varepsilon$ o al revés). Cada palabra ha de contener ambas subcadenas al menos una vez. Por ejemplo, $ccaaaaa\varepsilon$, $aabacbcccb\varepsilon$, $accaaab\varepsilon$, $ccbaabaab\varepsilon$ y $accbaaaabcc\varepsilon$ pertenecen a L_{42}. En cambio $bacbcc\varepsilon$ no pertenece a L_{42} porque no contiene la subpalabra $aa\varepsilon$.

43. L_{43} (0,050 puntos) Dar la definición formal del lenguaje L_{43} formado por las palabras que no contienen la subpalabra $aa\varepsilon$. Por ejemplo, $cabbccaba\varepsilon$, $cccabbbb\varepsilon$, $cccc\varepsilon$, ε y $accbbbabab\varepsilon$ pertenecen a L_{43}.

44. L_{44} (0,100 puntos) Dar la definición formal del lenguaje L_{44} formado por las palabras que en caso de contener b's, todas las b's están juntas. Por ejemplo, $ccaaaaa\varepsilon$, $aabbbcccca\varepsilon$, $ccc\varepsilon$, $bbaccaaa\varepsilon$, ε, $bbbb\varepsilon$ y $ccbbb\varepsilon$ pertenecen a L_{44}. En cambio $bacbcc\varepsilon$ no pertenece a L_{44} porque las b's no están juntas.

45. L_{45} (0,050 puntos) Dar la definición formal del lenguaje L_{45} formado por las palabras cuya longitud es al menos 2 y que terminan con el mismo símbolo con el que empiezan. Por ejemplo, $aabacbca\varepsilon$, $bcb\varepsilon$, $babb\varepsilon$ y $ccccc\varepsilon$ pertenecen a L_{45}. En cambio,

$cbbb\varepsilon$ no pertenece a L_{45} porque no acaba con el mismo símbolo con el que empieza y c no pertenece a L_{45} porque su longitud es menor que 2.

46. L_{46} (0,050 puntos) Dar la definición formal del lenguaje L_{46} formado por las palabras que se obtienen concatenando la palabra $ab\varepsilon$ todas las veces que se quiera. Por ejemplo, $ababab\varepsilon$, $ab\varepsilon$ y ε pertenecen a L_{46}. En cambio, $aba\varepsilon$, $bababa\varepsilon$ y $cabc\varepsilon$ no pertenecen a L_{46}.

47. L_{47} (0,050 puntos) Dar la definición formal del lenguaje L_{47} formado por las palabras que contienen la subpalabra $aa\varepsilon$ o la subpalabra $cc\varepsilon$. Cada palabra ha de contener al menos una de las subcadenas al menos una vez. Por ejemplo, $ccaaaaaa\varepsilon$, $bacbcccb\varepsilon$, $acaaab\varepsilon$, $cccc\varepsilon$, $ccba\varepsilon$ y $aabccccb\varepsilon$ pertenecen a L_{47}. En cambio $bacbca\varepsilon$ no pertenece a L_{47} porque no contiene ni la subpalabra $aa\varepsilon$ ni la subpalabra $cc\varepsilon$.

48. L_{48} (0,050 puntos) Dar la definición formal del lenguaje L_{48} formado por las palabras que empiezan por a, terminan en b y contienen al menos una c. Por ejemplo, $accaaaab\varepsilon$, $aabbcbccbb\varepsilon$, $acb\varepsilon$ y $aaccbaccb\varepsilon$ pertenecen a L_{48}. En cambio, ε, $bacbcc\varepsilon$ y $bbbb\varepsilon$ no pertenecen a L_{48}.

49. L_{49} (0,025 puntos) Dar la definición formal del lenguaje L_{49} formado por las palabras cuya longitud es mayor que tres y contienen una a en la tercera posición. Por ejemplo, $aaaa\varepsilon$, $ccab\varepsilon$, $cbabbbaac\varepsilon$, $ccabcbaaaa\varepsilon$ y $bcacccc\varepsilon$ pertenecen a L_{49}. Pero ε, $aa\varepsilon$, $aaa\varepsilon$, $ba\varepsilon$, $aabbca\varepsilon$ y $bba\varepsilon$ no pertenecen a L_{49}.

50. L_{50} (0,075 puntos) Dar la definición formal del lenguaje L_{50} formado por las palabras que empiezan por a, terminan en b, contienen una única c, entre la a inicial y la única c hay una serie de b's (cero o más) pero ninguna a y entre la única c y la b final hay una serie de a's (cero o más) pero ninguna b. Por ejemplo, $abbbcaab\varepsilon$, $acb\varepsilon$, $acaaab\varepsilon$ y $abbbbcb\varepsilon$ pertenecen a L_{50}. En cambio, $abba\varepsilon$, ε, $abbcaba\varepsilon$, $abbcac\varepsilon$, $acbbb\varepsilon$, $aaa\varepsilon$ y $ab\varepsilon$ no pertenecen a L_{50}.

51. L_{51} (0,050 puntos) Dar la definición formal del lenguaje L_{51} formado por las palabras que contienen al principio una serie de a's (una o más), luego una serie de b's (al menos una b) y al final una serie de c's (justo el mismo número de c's que de a's). Por ejemplo, $aabcc\varepsilon$, $bbbb\varepsilon$, $b\varepsilon$, $abbc\varepsilon$ y $aabbbcc\varepsilon$ pertenecen a L_{51}. En cambio, $bc\varepsilon$, $ac\varepsilon$, ε, $aaccbbb\varepsilon$ y $aaabbb\varepsilon$ no pertenecen a L_{51}.

52. L_{52} (0,075 puntos) Dar la definición formal del lenguaje L_{52} formado por las palabras que contienen la subpalabra $abc\varepsilon$ al principio o al final (o en ambos sitios). La subpalabra $abc\varepsilon$ puede aparecer o no aparecer en diferentes posiciones intermedias de la palabra. Por ejemplo, $abcaaaa\varepsilon$, $abc\varepsilon$, $accaaabc\varepsilon$, $abcbbbabc\varepsilon$ y $abccabcaaa\varepsilon$ pertenecen a L_{52}. En cambio, ε, $a\varepsilon$ y $bacbcc\varepsilon$ no pertenecen a L_{52}.

53. L_{53} (0,025 puntos) Dar la definición formal del lenguaje L_{53} formado por las palabras que no pertenecen a L_{52}.

54. L_{54} (0,075 puntos) Dar la definición formal del lenguaje L_{54} formado por las palabras que en caso de contener b's, no contienen c's. Por ejemplo, $ccaaaaa\varepsilon$, $aabbba\varepsilon$, $ccc\varepsilon$, $aaaa\varepsilon$, ε, $bbbb\varepsilon$ y $acaac\varepsilon$ pertenecen a L_{54}. En cambio, $bacbcc\varepsilon$ no pertenece a L_{54}.

55. L_{55} (0,075 puntos) Dar la definición formal del lenguaje L_{55} formado por las palabras que empiezan por a y luego no contienen ninguna c pero contienen al menos dos b's, o empiezan por a y luego solo contienen c's. Por ejemplo, $abb\varepsilon$, $aababa\varepsilon$, $aabaaab\varepsilon$, $aababab\varepsilon$ y $accccc\varepsilon$ son palabras de L_{55} mientras que ε, $aabbcb\varepsilon$, $caacbb\varepsilon$, $cccc\varepsilon$ y $bbc\varepsilon$ no son palabras de L_{55}.

56. L_{56} – Lenguaje formado por palabras que contienen por lo menos un símbolo y en las posiciones pares siempre hay una a y en las impares siempre una b. Por ejemplo, $babab\varepsilon$, $b\varepsilon$ y $bababa\varepsilon$ son palabras de L_{56} mientras que ε, $aabbcb\varepsilon$, $caacbb\varepsilon$, $cccc\varepsilon$ y $bbc\varepsilon$ no son palabras de L_{56}.

57. L_{57} – Lenguaje formado por las palabras que tienen longitud par y en las posiciones pares siempre hay una a y en las impares siempre una b. Por ejemplo, ε, $baba\varepsilon$, $ba\varepsilon$ y $bababa\varepsilon$ son palabras de L_{57} mientras que $aabbcb\varepsilon$, $caacbb\varepsilon$, $cccc\varepsilon$, $babab\varepsilon$ y $bbc\varepsilon$ no son palabras de L_{57}.

3.10.

Símbolos griegos

Es habitual utilizar símbolos del alfabeto griego para representar fórmulas de la lógica matemática y también para denominar funciones. Debido a ello, en la tabla 3.10.1 de la página 108, se muestran los símbolos griegos que se pudieran utilizar.

Alfabeto griego moderno

	Mayúscula	minúscula	Nombre (en castellano)
1	A, A	$\alpha, \boldsymbol{\alpha}$	alfa
2	B, B	$\beta, \boldsymbol{\beta}$	beta
3	$\Gamma, \boldsymbol{\Gamma}$	$\gamma, \boldsymbol{\gamma}$	gamma
4	$\Delta, \boldsymbol{\Delta}$	$\delta, \boldsymbol{\delta}$	delta
5	E, E	$\epsilon, \boldsymbol{\epsilon}, \varepsilon, \boldsymbol{\varepsilon}$	épsilon
6	Z, Z	$\zeta, \boldsymbol{\zeta}$	dseta
7	H, H	$\eta, \boldsymbol{\eta}$	eta
8	$\Theta, \boldsymbol{\Theta}$	$\theta, \boldsymbol{\theta}, \vartheta$	ceta
9	I, I	$\iota, \boldsymbol{\iota}$	iota
10	K, K	$\kappa, \boldsymbol{\kappa}, \varkappa, \boldsymbol{\varkappa}$	kappa
11	$\Lambda, \boldsymbol{\Lambda}$	$\lambda, \boldsymbol{\lambda}$	lambda
12	M, M	$\mu, \boldsymbol{\mu}$	mu
13	N, N	$\nu, \boldsymbol{\nu}$	nu
14	$\Xi, \boldsymbol{\Xi}$	$\xi, \boldsymbol{\xi}$	ksi
15	O, O	o, o	ómicron
16	$\Pi, \boldsymbol{\Pi}$	$\pi, \boldsymbol{\pi}, \varpi$	pi
17	P, P	$\rho, \boldsymbol{\rho}, \varrho, \boldsymbol{\varrho}$	ro
18	$\Sigma, \boldsymbol{\Sigma}$	$\sigma, \boldsymbol{\sigma}, \varsigma$	sigma
19	T, T	$\tau, \boldsymbol{\tau}$	tau
20	Υ, Y	$\upsilon, \boldsymbol{\upsilon}$	ípsilon
21	$\Phi, \boldsymbol{\Phi}$	$\phi, \boldsymbol{\phi}, \varphi, \boldsymbol{\varphi}$	fi
22	X, X	$\chi, \boldsymbol{\chi}$	ji
23	$\Psi, \boldsymbol{\Psi}$	$\psi, \boldsymbol{\psi}$	psi
24	$\Omega, \boldsymbol{\Omega}$	$\omega, \boldsymbol{\omega}$	omega

Letras griegas no pertenecientes al alfabeto griego moderno

	Mayúscula	minúscula	Nombre (en castellano)
25	$\digamma$	$\boldsymbol{\digamma}$	digamma
26	Q	$\boldsymbol{\mathsf{q}}$	qoppa
27	$\boldsymbol{\mathsf{\varkappa}}$	$\boldsymbol{\mathsf{\varkappa}}$	koppa
28	$\boldsymbol{\mathsf{\varkappa}}$	$\boldsymbol{\mathsf{\varkappa}}$	sampi
29	$\boldsymbol{\mathsf{\varsigma}}$	$\boldsymbol{\mathsf{\varsigma}}$	estigma

Tabla 3.10.1. Símbolos griegos.