TEMA 1:

Lenguajes Formales

Autómatas y Lenguajes Formales.
Grado en Informática (2° curso).

Dpto. de Ingeniería de la Información y las Comunicaciones.

UNIVERSIDAD DE MURCIA

Índice general

1. Lenguajes naturales y lenguajes formales

Antes de comenzar a definir los conceptos más elementales sobre los lenguajes formales, vamos a indicar algunas características de los lenguajes naturales que los hacen ser complejos y motivan el uso de lenguajes formales más simples en Informática.

Los **lenguajes naturales** (idiomas, lenguas) *se usan para la comunicación entre humanos* y algunas características que presentan son las siguientes:

- Tienen mucha *expresividad y gran variedad* de componentes lingüísticos (nombres, verbos, adjetivos, pronombres, etc.) y de formas de componerlos para formar oraciones.
- Debido a la riqueza del lenguaje natural, las reglas gramaticales para el buen uso morfológico y sintáctico de un idioma son *difíciles de establecer con precisión*. Además, las reglas son *poco rígidas*, pueden cambiar con el tiempo y suele ocurrir que aunque alguien se exprese en un idioma incorrectamente eso no impide que se entienda el mensaje que se comunica o se recibe.
- En los lenguajes naturales una misma palabra u oración puede tener significados distintos, y esto recibe el nombre de *ambigüedad*.
- La ambigüedad hace que la semántica de la oración *dependa del contexto* en que se emite el mensaje. La forma o sintaxis de la oración no impone un significado único.

¿Cómo hacer que una máquina 'comprenda' el lenguaje natural?

Este es un campo de la Inteligencia Artificial que se denomina *Procesamiento del Lenguaje Natural* y tiene un gran interés para el desarrollo de aplicaciones en las que se pretende conseguir una comunicación humano-ordenador más próxima a la forma en que las personas se comunican entre sí (por ejemplo, en interfaces de consulta a la web, a bases de datos o a sistemas automáticos de atención al cliente). También resulta interesante el procesamiento del lenguaje natural para traducción automática de un idioma a otro.

En relación con el Procesamiento del Lenguaje Natural, se plantean también otros problemas complejos como el reconocimiento de la escritura o del habla para traducir los mensajes a un formato digital adecuado, pero sobre todo la dificultad reside en hacer que una máquina (entiéndase aplicación software más hardware necesario para ejecutarla) *interprete correctamente* el significado de los mensajes que recibe para que sean procesados de la forma adecuada. Actualmente se consiguen resultados muy buenos con los grandes modelos del lenguaje y herramientas como ChatGPT, pero no se puede garantizar la precisión en la interpretación del lenguaje natural. Además, estos sistemas requieren mucha capacidad de cómputo.

¿Por qué se requiere el uso de lenguajes formales en Informática?

Las características anteriores del lenguaje natural, que son adecuadas para una comunicación fluida entre humanos, son causa de dificultades a la hora de asegurar que la

comunicación humano-ordenador sea fiable y eficiente, debido sobre todo al problema de la ambigüedad.

En determinadas áreas, como la programación, se necesita una *precisión total* para dar órdenes a un ordenador; por tanto, debe *evitarse la ambigüedad* para que la sintaxis determine de forma única la semántica, con objeto de que la interpretación y ejecución de las órdenes sea correcta. Se requiere además que el ordenador proporcione una *respuesta lo más rápida posible* a las instrucciones que recibe y para ello es necesario que se minimice el tiempo que el ordenador emplea en analizar las órdenes antes de pasar a ejecutarlas.

Debido a los requerimientos anteriores de precisión y eficiencia, en Informática se necesitan **lenguajes formales** más simples que el lenguaje natural, para definir los comandos del sistema operativo y las instrucciones de los lenguajes de programación, para describir con precisión el formato de entrada a un programa, para describir comandos de formateo de texto o realizar conversiones de formato entre archivos, etc.

El uso de lenguajes formales evita la dificultad computacional que supone el procesamiento del lenguaje natural.

Un LENGUAJE FORMAL no es más que un conjunto de cadenas que se caracterizan por cumplir una determinada propiedad que puede describirse de manera precisa, por ejemplo mediante reglas gramaticales o de sintaxis.

Ejemplo 1 *Ejemplos de lenguajes formales.*

- El **lenguaje de la lógica proposicional**. Las cadenas de este lenguaje se llaman *fórmulas proposicionales* o proposiciones. Las fórmulas se escriben según unas reglas sintácticas rígidas que permiten, entre otras cosas, distinguir claramente las cadenas que son fórmulas bien formadas (sintácticamente correctas) de las que no los son.
  Ej. La cadena "$v_1 \vee \neg v_2 \wedge v_3$" es una fórmula del lenguaje de la lógica proposicional, mientras que "$\vee v_1 \wedge v_2$" no tiene la forma correcta para considerarse una fórmula. Una vez que se sabe que "$v_1 \vee \neg v_2 \wedge v_3$" es una fórmula sintácticamente correcta, puede pasarse a evaluarla dando valores booleanos a las variables v_1, v_2, v_3, teniendo en cuenta las reglas de evaluación (tablas de verdad) de los operadores lógicos (conectivas $\vee, \wedge, \neg$). Para cada asignación particular de valores a las variables, el valor de verdad resultante (que equivale al significado de la fórmula) es único, no hay ambigüedad en el resultado o, al menos, puede evitarse con facilidad estableciendo reglas de precedencia y asociatividad de operadores. El hecho de que el lenguaje de la lógica sea tan formal facilita el proceso de razonamiento dentro del sistema de la lógica y también permite en muchos casos mecanizar el razonamiento.
- El **lenguaje de programación C.** Las cadenas de este lenguaje son los programas sintácticamente correctos en C. No es lo mismo un algoritmo que un programa en C. En el algoritmo no se tiene en cuenta una sintaxis estricta para escribir las instrucciones, para el tipo de las variables o para usar sentencias de entrada/salida, porque lo que interesa es mostrar la esencia de los cálculos que se necesitan para resolver un problema. El lenguaje algorítmico abstracto puede considerarse semi-formal, porque es una mezcla de formalismos y lenguaje natural; el lenguaje algorítmico en

pseudocódigo es más formal, pero para que un ordenador lo entienda es necesario implementar los algoritmos en programas escritos en un lenguaje formal como C, para que pueda ser compilado y ejecutado por la máquina.

- El **lenguaje de direcciones de correo electrónico.** Es un lenguaje más simple que los anteriores. Está formado por cadenas que respetan la sintaxis de las direcciones de correo.
  Ej. "unalumno@um.es" es una cadena del lenguaje, pero "unalumno@es" no lo es, porque le falta la subcadena que identifica al dominio del servidor de correo.

Especificación y procesamiento de lenguajes formales

La mayoría de lenguajes formales de interés en Informática contienen potencialmente un número infinito de cadenas y resulta fundamental encontrar una *descripción computacional* del lenguaje, que es aquella que especifica formalmente (de forma precisa, sin ambigüedad) el lenguaje mediante una *expresión finita*, codificable y aplicable en tareas de procesamiento de cadenas del lenguaje.

Los *formalismos y modelos teóricos* más comunes para la especificación formal y el procesamiento de lenguajes formales, que estudiaremos en los distintos temas de la asignatura, son las *expresiones regulares, las gramáticas y los autómatas*. También se usan estos formalismos en determinadas tareas relacionadas con el procesamiento del lenguaje natural. De hecho las gramáticas tienen su origen en los trabajos de Noam Chomsky en lingüística.

Introducimos a continuación un ejemplo de gramática. Una gramática es un conjunto finito de reglas con las que se puede describir la sintaxis (estructura) de las cadenas de un lenguaje formal. Con las reglas se pueden generar cadenas del lenguaje y también se usan las reglas en algoritmos que comprueban si una cadena de símbolos tiene el formato adecuado para que se considere dentro del lenguaje especificado por la gramática, es decir, para comprobar si es sintácticamente correcta. **Las gramáticas son de especial interés para describir formalmente la sintaxis de los lenguajes de programación y para construir compiladores.**

Ejemplo 2 *Gramática para un lenguaje de programación simple.*

Supongamos que tenemos un lenguaje de programación simple y describimos informalmente los programas del siguiente modo: "un PROGRAMA comienza por la palabra clave **begin**, va seguido de un BLOQUE DE SENTENCIAS (debe haber al menos una SENTENCIA del bloque) y termina por la palabra clave **end**. Una SENTENCIA puede ser una sentencia de ASIGNACIÓN o una sentencia condicional tipo IF o una sentencia iterativa tipo WHILE. Una sentencia de ASIGNACIÓN se forma ..., etc." Esta descripción narrada de la sintaxis se especifica formalmente mediante las siguientes reglas gramáticales (se incluyen sólo las primeras reglas):

⟨programa⟩ → **begin** ⟨bloque-sentencias⟩ **end**
⟨bloque-sentencias⟩ → ⟨sentencia⟩ | ⟨sentencia⟩⟨bloque-sentencias⟩
⟨sentencia⟩ → ⟨sent-if⟩ | ⟨sent-while⟩ | ⟨sent-asig⟩
⟨sent-asig⟩ → ...
...

Lo que aparece entre ángulos son variables de la gramática y representan porciones de las cadenas del lenguaje, en este caso de los programas. Por ejemplo, la variable ⟨programa⟩ representa a un programa completo y la variable ⟨sentencia⟩ representa a una sentencia de un programa. Cada regla de la gramática tiene una variable en la parte izquierda de la flecha y una o varias expresiones separadas por | en parte derecha, que describen en qué consiste la parte de un programa representada por la variable de la izquierda.

Para el diseño e implementación de una aplicación compleja de procesamiento de cadenas de un lenguaje formal, como es el caso de un compilador, se usan modelos teóricos, algoritmos y herramientas que combinan expresiones regulares, gramáticas y autómatas en las diferentes etapas de construcción de la aplicación. Sin estos métodos formales sería muy costoso el desarrollo de nuevos lenguajes de programación o de versiones actualizadas de lenguajes ya existentes. A lo largo de la asignatura veremos distintos tipos de autómatas y gramáticas y varias aplicaciones de procesamiento de cadenas de lenguajes formales basados en estos formalismos. La asignatura de Compiladores puede considerarse una continuación natural de la asignatura de Autómatas y Lenguajes Formales, donde se aplican los conceptos teóricos que vamos a introducir.

Aplicaciones de los autómatas y lenguajes formales

Desde su nacimiento en los años 40 del pasado siglo, la Teoría de Autómatas y Lenguajes Formales ha encontrado **aplicación en campos muy diversos**, no sólo en el ámbito de los lenguajes de programación. Algunos de ellos son:

- Búsqueda de patrones y análisis de secuencias de patrón regular.
- Conversión de formato, compiladores e intérpretes.
- Procesamiento del lenguaje natural.
- Modelado de protocolos de comunicación.
- Modelado de sistemas de eventos discretos y de control.
- Diseño y verificación de circuitos digitales secuenciales y automatismos industriales.

2. Alfabetos y cadenas. Operaciones con cadenas

Definición 1 (definiciones básicas)

- Un ALFABETO (o *vocabulario*) V es un conjunto finito y no vacío de elementos llamados *símbolos*. Ej. $V = \{a, b, c\}$ es un alfabeto de tres símbolos representados por las letras a, b, c. Para indicar que un símbolo forma parte de un alfabeto se usa la notación habitual de pertenencia de un elemento a un conjunto. Ej. $a \in V$.

  Por defecto usaremos la letra V para designar a un alfabeto. Se puede usar V con subíndice opcional aclaratorio de los símbolos que contiene o incluso un nombre más significativo. Un alfabeto como $V = \{a, b, c\}$ es *abstracto* en el sentido de que sus símbolos en principio no tienen significado especial. Estos alfabetos cortos y abstractos los usaremos en ejercicios teóricos por comodidad, porque una vez que se saben resolver problemas con un alfabeto abstracto se pueden resolver problemas de tipo más práctico. Otros alfabetos son menos abstractos, en el sentido de que podemos darle una *interpretación común*. Por ejemplo:

  $V_{dig} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ es el alfabeto de los dígitos decimales.
  $V_{lat} = \{a, b, \ldots, z, A, \ldots, Z\}$ es el alfabeto de las letras latinas.
  $V_{bin} = \{0, 1\}$ es el alfabeto de dígitos binarios.
  ASCII es el alfabeto formado por todos los caracteres ASCII.

- Una CADENA (o *palabra*) es una secuencia finita de símbolos de cierto alfabeto V. En general, usaremos las letras w, x, y, z como variables de cadena, para referirnos a *cadenas arbitrarias*. Ej. considerando el alfabeto $\{a, b, c\}$ tenemos que $aa, abb, bcacc$ son cadenas formadas por símbolos de ese alfabeto, porque en ellas sólo aparecen los símbolos a, b, c y se tiene que $a, b, c \in \{a, b, c\}$. Obsérvese que a, b, c además de ser símbolos pueden considerarse también cadenas.

- La LONGITUD DE UNA CADENA w se denota como $|w|$ y es el número de símbolos que aparecen en w. Ej. $|bcacc| = 5, |a| = 1$.

- A la cadena que no tiene sí mbolos la llamaremos CADENA VACÍ A y se denota por la letra griega λ (en algunos libros se usa ϵ). Se tiene que $|\lambda| = 0$.La cadena vacía no es un símbolo, así que es incorrecto afirmar que $\lambda \in V$, sea cual sea el alfabeto V.

- V^* denota el conjunto de las cadenas de cualquier longitud que se forman con los símbolos del alfabeto V, incluida la cadena vacía λ, y se llama LENGUAJE UNIVERSAL con alfabeto V. Siempre se tiene que $\lambda \in V^*$, con independencia de los símbolos del alfabeto V. Ej. considerando que $V = \{a, b, c\}$ entonces $V^* = \{a, b, c\}^*$ es el lenguaje universal con alfabeto $\{a, b, c\}$ que contiene todas las cadenas formadas por a's o b's o c's y también la cadena vacía; así $\lambda, a, abb, c, bcacc$ son cadenas de $\{a, b, c\}^*$, que expresamos formalmente como: $\lambda, a, abb, c, bcacc \in \{a, b, c\}^*$.

- V^+ es el conjunto de las cadenas con alfabeto V de longitud mayor o igual que 1 (no incluye λ). Se cumple que $V^* = V^+ \cup \{\lambda\}$.

Ejemplo 3 *Consideremos el alfabeto de dígitos decimales* $V_{dig} = \{0, 1, 2, \ldots, 9\}$.

- *En el lenguaje universal* V_{dig}^* *están todas las cadenas formadas por concatenación de símbolos que representan dígitos decimales, junto con la cadena vacía* λ. *Las cadenas de* V_{dig}^+ *(no incluye* λ*), podemos interpretarlas como números naturales en notación decimal.*
- *Es* ***incorrecto*** *escribir* $245 \in V_{dig}$, *porque los elementos del alfabeto* V_{dig} *son símbolos, no cadenas. Es* ***correcto*** *escribir* $245 \in V_{dig}^*$ *y también* $2 \in V_{dig}^*$, *porque* 2, *además de ser un símbolo del alfabeto (*$2 \in V_{dig}$*), es también una cadena de longitud* 1 *del lenguaje universal* V_{dig}^*.

No debemos pensar que los símbolos son los caracteres con que se escriben las palabras de un texto o los números. Son más que eso:

> **Los símbolos** de un alfabeto son entidades *abstractas* que se usan para representar (*simbolizar*) **componentes indivisibles** de un lenguaje formal (no se pueden 'partir' en otros símbolos) y que **se combinan para formar cadenas o secuencias** de muy diverso tipo.

Ejemplo 4 *Con el alfabeto* $V = \{A, C, G, T\}$ *se pueden representar las bases nitrogenadas que se conectan en una cadena de ADN, a saber:* ***A****denina,* ***C****itosina,* ***G****uanina,* ***T****imina. Ciertas cadenas que se forman con estos símbolos, como* $GATTACA$*, representan* ***secuencias de ADN****. En Bioinformática es de vital interés analizar secuencias de ADN y puede hacerse en parte aplicando diversas técnicas que proporciona la Teoría de Autómatas y Lenguajes Formales.*

A veces, los símbolos se usan para representar *eventos* que suponen un cambio de estado en un sistema y opcionalmente la realización de alguna acción. Se puede definir un alfabeto con un símbolo para representar a cada evento y las cadenas serían *secuencias de eventos* que se producen en cierto intervalo de tiempo.

Ejemplo 5 *Podemos usar nombres más significativos para los símbolos de eventos. Ej. el alfabeto* $\textsc{EvBoton} = \{aceptar, cancelar, cerrar\}$ *contiene tres símbolos que representan eventos que se producen por la pulsación de botones en una ventana de una interfaz gráfica. El símbolo 'aceptar' representa un único evento y los caracteres individuales del nombre del símbolo no tienen sentido de forma aislada. Si hay más botones que generen eventos en la aplicación se ampliaría este alfabeto de símbolos de evento. Una cadena con estos símbolos representa una secuencia de eventos de pulsación de botones.*

2.1. Concatenación de cadenas

La operación fundamental con cadenas de cierto lenguaje universal V^* es la concatenación. Es una operación binaria que denotamos con el operador $\circ$ y se define formalmente

como:

$$\boxed{\begin{array}{c} \circ : V^* \times V^* \longrightarrow V^* \\ x \circ y \mapsto xy \end{array}}$$

La definición anterior indica que la concatenación se aplica a dos cadenas $x, y \in V^*$ y produce como resultado la cadena xy con los símbolos de la primera cadena x seguidos de los símbolos de la segunda cadena y.

Ejemplo 6 *Dado un alfabeto $V = \{a, b\}$ y las cadenas $x = abb$, $y = aaaa$, entonces $x \circ y = xy = abbaaaa$.*

PROPIEDADES BÁSICAS DE LA CONCATENACIÓN:

- **Propiedad asociativa**: $\forall\, x, y, z \in V^* :\ x \circ (y \circ z) = (x \circ y) \circ z$
- **Elemento identidad λ**: $\forall\, x \in V^* :\ \lambda \circ x = x \circ \lambda = x$

Definición 2 (subcadena, prefijo y sufijo)

- Una cadena z es **subcadena** de otra cadena $w \overset{def}{\Longleftrightarrow} \exists\, x, y : w = x \circ z \circ y$
- Una cadena z es un **prefijo** de $w \overset{def}{\Longleftrightarrow} \exists\, x :\ w = z \circ x$
- Una cadena z es un **sufijo** de $w \overset{def}{\Longleftrightarrow} \exists\, x :\ w = x \circ z$

Nota: cuando esté claro por el contexto se puede dejar implícito el operador de concatenación '$\circ$', por acortar. Por ejemplo: en la definición anterior de subcadena se puede escribir directamente que z es subcadena de w si y sólo si , por definición, $\exists\, x, y : w = xzy$.

Ejemplo 7 *Dada la cadena $w = abab$, tenemos que:*

λ, ba y $abab$ son subcadenas de w
ab es prefijo, sufijo y subcadena de w
b es un sufijo de w, pero no es prefijo de w

Nota:

- λ es subcadena, prefijo y sufijo de cualquier cadena
- Toda cadena w es subcadena, prefijo y sufijo de sí misma

2.2. Potencia de cadenas

Definición 3 *La* POTENCIA *n-ésima de una cadena es una operación unaria que consiste en concatenar la cadena consigo misma n veces. Formalmente, la potencia n-ésima de una cadena w, que se denota w^n, se define* ***recursivamente*** *(más concretamente, de forma inductiva porque depende del parámetro numérico n) como :*

1. *Regla base: para $n = 0$ se tiene que* $\boxed{w^0 = \lambda}$
2. *Regla recursiva: si $n > 0$ entonces* $\boxed{w^n = w \circ w^{n-1} = w^{n-1} \circ w}$

Ejemplo 8 *Dada la cadena $w = aba$, vamos a obtener w^3 según la definición recursiva, realizando primero una* ***evaluación ascendente*** *de w^3 (empezando por el caso base w^0 y resolviendo cada potencia mediante concatenación usando el resultado de la potencia anterior):*

$$w^0 = (aba)^0 = \lambda$$
$$w^1 = w \circ w^0 = aba \circ \lambda = aba$$
$$w^2 = w \circ w^1 = aba \circ aba = abaaba$$
$$w^3 = w \circ w^2 = aba \circ abaaba = abaabaaba$$

La ***evaluación descendente*** *de w^3 (desarrollando según la definición hasta llegar al caso base y después resolviendo) es:*

$$w^3 = (aba)^3 \overset{def}{=} aba \circ (aba)^2 = aba \circ (aba \circ (aba)^1) = aba \circ (aba \circ (aba \circ (aba)^0)) \overset{resuelve}{=}$$
$$aba \circ (aba \circ (aba \circ \lambda)) = aba \circ (aba \circ aba) = aba \circ (abaaba) = abaabaaba$$

Nota:

- **La potencia tiene mayor precedencia que la concatenación**. Eso significa que hay que usar paréntesis cuando sea oportuno para alterar el convenio de precedencia de operadores, como en las operaciones aritméticas. Ej. al escribir w^2 se entiende que la cadena resultante es $w \circ w = ww$, pero si se escriben los símbolos concretos de w hay que encerrar los símbolos entre paréntesis. Ej. si $w = 234$ entonces $w^2 = (234)^2 = 234 \circ 234 = 234234$, mientras que si no ponemos los paréntesis tenemos que $234^2 = 23 \circ (4)^2 = 23 \circ 44 = 2344$.
- w^0 **siempre es** λ, independientemente de cómo sea la cadena w. Ej. considerando que 234 es una cadena de símbolos de dígitos decimales, esto es: $234 \in V_{dig}^*$, entonces $(234)^0 = \lambda$. No es cierto que $(234)^0 = 1$. Ese sería el resultado en caso de interpretar 234 como un número y aplicar la operación de potencia numérica.

2.3. Reflexión de cadenas

Informalmente, dada una cadena w, la *cadena refleja* w^R se obtiene 'dándole la vuelta' a los símbolos de w. Por ejemplo, si $w = aab$ entonces $w^R = baa$. Se puede definir

formalmente la cadena refleja de manera recursiva teniendo en cuenta que la cadena sea vacía (caso base) o que tenga al menos un símbolo:

1. REGLA BASE: para $w = \lambda$ se tiene que $\xrightarrow{def}$ $\boxed{\lambda^R = \lambda}$
2. REGLA RECURSIVA: si $w = a \circ z$ con $a \in V, z \in V^*$ entonces $\boxed{(a \circ z)^R = z^R \circ a}$

Nota: **La reflexión tiene mayor precedencia que la concatenación y la misma que la potencia**.

Ejemplo 9 *Dada la cadena aab entonces* $(aab)^R = baa$ *y se puede comprobar a partir de la definición recursiva haciendo una evaluación descendente de* $(aab)^R$*, teniendo en cuenta que* $abb = a \circ ab$*:*

$$(aab)^R = (a \circ ab)^R \overset{def}{=} (ab)^R \circ a = (a \circ b)^R \circ a = (b^R) \circ a \circ a = (\lambda^R) \circ b \circ a \circ a \overset{resolver}{=} baa$$

De forma más compacta omitiendo el operador de concatenación '$\circ$*' en todos los pasos de la evaluación recursiva sería:*

$$(aab)^R = (ab)^R a = (b^R)aa = (\lambda^R)baa = \lambda baa = baa$$

3. Descripción matemática de lenguajes

Recordamos que V^* denota al **lenguaje universal** con alfabeto V, que contiene a todas las cadenas de cualquier longitud formadas por símbolos del alfabeto V y la cadena vacía λ. Seleccionando distintos tipos de cadenas del lenguaje universal se forman distintos lenguajes. En general, se dice que un LENGUAJE con alfabeto V es un conjunto de cadenas de símbolos del alfabeto V. Más formalmente se dice que:

Definición 4 *Un lenguaje* L *con alfabeto* V *es un subconjunto del lenguaje universal* V^* *y se denota como* $L \subseteq V^*$*.*

Nota: El **lenguaje vacío** $\varnothing$ y el propio alfabeto V pueden considerarse lenguajes, por ser subconjuntos de V^*. No es lo mismo $\varnothing$ que $\{\lambda\}$, porque el lenguaje $\{\lambda\}$ contiene la cadena vacía y el lenguaje $\varnothing$ no contiene ninguna cadena.

Entre todos los lenguajes incluidos en el lenguaje universal (hay un número infinito no numerable de ellos, tantos como subconjuntos de V^*) nos interesan los **lenguajes** FORMALES, que son aquellos cuyas cadenas cumplen cierta propiedad que puede describirse o especificarse de manera precisa (formal), a partir de una descripción informal o en lenguaje natural. Al ser los lenguajes formales conjuntos de cadenas podemos describirlos formalmente como se hace habitualmente para definir conjuntos en matemáticas. Entre los diversos tipos de definiciones o descripciones matemáticas de conjuntos/lenguajes destacamos en este tema las **definiciones por extensión** y las **definiciones por comprensión** (también llamadas **definiciones por intensión**), que explicamos a continuación.

Definición de un lenguaje por extensión

Consiste simplemente en enumerar las cadenas del lenguaje entre llaves y separadas por comas. Es una descripción que sólo sirve para lenguajes finitos.

Nota: cuando tenemos un lenguaje finito L, la **cardinalidad** de L es el número de elementos (cadenas) de L, que se expresa como $|L|$ (no confundir con $|w|$, que es la longitud o número de símbolos de la cadena w).

Ejemplo 10 *Consideramos la siguiente* ***descripción informal*** *(en lenguaje natural) de un lenguaje que llamamos* BinLon2*: "el lenguaje* BinLon2 *está formado por todas las cadenas de dígitos binarios de longitud 2". El alfabeto de este lenguaje es* $\{0,1\}$ *y el lenguaje universal es* $\{0,1\}^*$*. La* ***descripción formal por extensión*** *del lenguaje es*

$$\textit{BinLon2} = \{00, 01, 10, 11\}$$

Se tiene que $\textit{BinLon2} \subseteq \{0,1\}^*$*. Está claro que* $|\textit{BinLon2}| = 4$*, porque hay 4 cadenas en el lenguaje* BinLon2*. Además, para toda cadena* $w \in \textit{BinLon2}$ *se tiene que* $|w| = 2$*.*

Definición de un lenguaje por comprensión (o intensión)

Consiste en describir de forma precisa y sin ambigüedad el tipo de cadenas que forman parte del lenguaje usando alguna propiedad que las caracterice (llamada **propiedad de pertenencia** al lenguaje), que se especifica mediante una o varias condiciones o predicados lógico-matemáticos, siguiendo el siguiente ***esquema de definición por comprensión***:

$$\boxed{L = \{w \in V^* \mid P(w)\}}$$

L es el **nombre** del lenguaje. Usaremos la letra L con o sin subíndices, las primeras letras mayúsculas $A, B, C \ldots$ o nombres más significativos en mayúscula para nombrar a los lenguajes. La **descripción** del lenguaje es la expresión a la derecha del signo igual. El nombre del lenguaje permite hacer referencia posterior al lenguaje sin tener que usar la descripción completa. La definición consiste en un **preámbulo** (parte izquierda del símbolo 'tal que' $\mid$) donde se establece que las cadenas se seleccionan en el ámbito (*universo*) de cierto lenguaje universal V^*. De esta forma queda claro el alfabeto que se usa en el lenguaje. En el **cuerpo** de la definición (parte derecha de $\mid$) se describe la propiedad de pertenencia P. La expresión $\mathbf{P(w)}$ significa que la cadena w cumple la propiedad P que caracteriza a las cadenas del lenguaje L. Variando la propiedad P y el lenguaje universal V^* se obtendrían distintos lenguajes. Esta descripción por comprensión es adecuada para lenguajes infinitos o finitos con un número extenso de cadenas.

Ejemplo 11 *Supongamos que tenemos un lenguaje llamado* BinIg *que consiste en (descripción informal): "cadenas de dígitos binarios que tienen igual número de ceros que de unos". Podemos proporcionar una* ***definición por comprensión*** *del lenguaje* BinIg *de la siguiente forma:*

$$\textit{BinIg} = \{w \in \{0,1\}^* \mid \textit{ceros}(w) = \textit{unos}(w)\}$$

La propiedad de pertenencia $P(w)$ en este caso se corresponde con el predicado:

$$ceros(w) = unos(w)$$

Se supone que $ceros(w)$ es una función que nos da el número de ceros de la cadena w y $unos(w)$ nos da el número de unos que aparecen en w. Las cadenas w que están en el lenguaje BinIG *son precisamente las cadenas para las que el predicado "$ceros(w) = unos(w)$" es cierto. A partir de la descripción informal puede que no quede claro si la cadena vacía pertenece a este lenguaje o no; se supone que la cadena λ tiene el mismo número de ceros que de unos, porque no tiene ninguno. Por la descripción formal queda claro que $\lambda \in$ BinIG porque $ceros(\lambda) = 0$ y $unos(\lambda) = 0$, luego $ceros(\lambda) = unos(\lambda)$ y por tanto λ cumple la propiedad de pertenencia a* BinIG*, por tanto, $\lambda \in$ BinIG.*

Algunas definiciones por comprensión en las que queda claro cual es el alfabeto del lenguaje (y por tanto el lenguaje universal de referencia) se pueden **simplificar dejando implícito el lenguaje universal**.

Ejemplo 12 *Si tenemos el lenguaje descrito como:*

$$A = \{a^n b \mid n \geq 0\} \tag{1}$$

damos por supuesto que el alfabeto del lenguaje A es $\{a, b\}$ y por tanto el lenguaje universal es $\{a, b\}^*$. Aquí la propiedad de pertenencia se expresa con el patrón $a^n b, n \geq 0$, que se reparte entre el preámbulo y el cuerpo de la definición e indica el formato de las cadenas que pertenecen al lenguaje A. En el patrón aparece el parámetro numérico n, que al variarlo permite describir a todas las palabras del lenguaje. Cuando $n = 0$ se tiene que $a^0 b = b$, así que $b \in \{a^n b \mid n \geq 0\}$ y para $n = 1, 2, 3 \ldots$ se tiene que $ab, aab, aaab, \ldots$, son cadenas del lenguaje $\{a^n b \mid n \geq 0\}$.
La definición en (1) es equivalente a esta otra en (2) donde se especifica explícitamente el lenguaje universal:

$$A = \{w \in \{a, b\}^* \mid w = a^n b \text{ y } n \geq 0\} \tag{2}$$

Cuando hay varias condiciones en el cuerpo de la definición, como es el caso de la definición en (2), pueden separarse las condiciones mediante comas o mediante la conjunción lógica $\wedge$, en lugar de 'y' en lenguaje natural (son preferibles las dos primeras opciones, por no confundir 'y' con una cadena):

$$A = \{w \in \{a, b\}^* \mid w = a^n b, n \geq 0\}, \text{ o bien } A = \{w \in \{a, b\}^* \mid w = a^n b \wedge n \geq 0\}$$

La descripción matemática de lenguajes es adecuada como paso previo para formalizar la resolución computacional de un problema de procesamiento de cadenas del lenguaje. En los siguientes temas veremos cómo 'traducir' una descripción matemática en una **descripción computacional** de un lenguaje mediante alguno de los formalismos (también de carácter matemático) de expresiones regulares, autómatas o gramáticas y cómo usar dichos formalismos en diversos algoritmos de procesamiento de cadenas.

4. Operaciones con lenguajes

Los lenguajes se pueden combinar mediante operaciones que producen como resultado otro lenguaje. Algunas de las operaciones con lenguajes son las mismas que se pueden hacer entre conjuntos cualesquiera y otras son operaciones que tienen que ver con las operaciones entre cadenas que hemos introducido en la sección 2. Las operaciones con lenguajes, aparte de ser usadas en demostraciones de resultados teóricos, pueden servir como herramienta para obtener descripciones formales de lenguajes complejos a partir de otros lenguajes más simples.

4.1. Operaciones heredadas de la teoría de conjuntos

Las operaciones habituales de la teoría de conjuntos son aplicables a los lenguajes, ya que los lenguajes son conjuntos de cadenas. Hacemos ahora un repaso de las operaciones de conjuntos que se ven en la asignatura de Fundamentos Lógicos, teniendo en cuenta que los elementos de los conjuntos que nosotros manejamos son siempre cadenas de símbolos de cierto alfabeto V. El "universo" de todas las cadenas de símbolos de V, que tenemos en cuenta en las operaciones, es el lenguaje universal V^*.

Definición 5 *Sean L_1 y L_2 dos lenguajes:*

- *La **unión** de L_1 y L_2, denotada $L_1 \cup L_2$, es el lenguaje formado por las cadenas de ambos lenguajes. Formalmente:*

$$L_1 \cup L_2 = \{w \mid w \in L_1 \text{ o } w \in L_2\}$$

  O mejor usando las disyunción lógica $\vee$, por no confundir 'o' con una cadena o un símbolo.

$$\boxed{L_1 \cup L_2 = \{w \mid w \in L_1 \vee w \in L_2\}}$$

- *La **intersección** de dos lenguajes L_1 y L_2, denotada $L_1 \cap L_2$, es el lenguaje que contiene a cadenas que pertenecen tanto a L_1 como a L_2. Formalmente:*

$$\boxed{L_1 \cap L_2 = \{w \mid w \in L_1 \wedge w \in L_2\}}$$

- *La **diferencia** de los lenguajes L_1 y L_2, denotada $L_1 - L_2$, es el lenguaje que contiene a cadenas que pertenecen a L_1 y no pertenecen a L_2:*

$$\boxed{L_1 - L_2 = \{w \mid w \in L_1 \wedge w \notin L_2\}}$$

- *El lenguaje **complementario** de L_1 (con respecto a cierto lenguaje universal V^*) es el lenguaje, denotado como $\overline{L_1}$, que contiene a todas las cadenas del lenguaje universal V^* que no están en L_1. Formalmente:* $\boxed{\overline{L_1} = V^* - L_1}$

Ejemplo 13 *Consideremos de nuevo el lenguaje*

$$\textsc{BinIg} = \{w \in \{0,1\}^* \mid ceros(w) = unos(w)\}$$

Sea ahora el lenguaje BinImpar *que contiene las cadenas que representan números binarios impares. Este lenguaje se puede describir por comprensión sin hacer referencia a ninguna propiedad numérica como:*

$$\textsc{BinImpar} = \{z1 \mid z \in \{0,1\}^*\}$$

O lo que es lo mismo (usando el lenguaje universal en el preámbulo):

$$\textsc{BinImpar} = \{w \in \{0,1\}^* \mid w = z1\}$$

La propiedad de pertenencia indica, por tanto, que la cadena w debe acabar en 1 (además de contener sólo símbolos 0/1). En la segunda definición se deduce que $z \in \{0,1\}^*$, por tanto no es necesario incluir esta condición en el cuerpo de la descripción. Si queremos describir al lenguaje BinIgImpar *formado por todas las cadenas binarias que representan números impares y además tienen el mismo número de ceros que de unos podemos tener en cuenta que*

$$\textsc{BinIgImpar} = \textsc{BinIg} \cap \textsc{BinImpar} = \{w \mid w \in \textsc{BinIg} \wedge w \in \textsc{BinImpar}\}$$

La anterior no es una definición explícita de BinIgImpar *porque se hace referencia a otros lenguajes. Considerando la propiedad que caracteriza la pertenencia de w a cada lenguaje, podemos dar una descripción por comprensión explícita de* BinIgImpar *como:*

$$\textsc{BinIgImpar} = \{w \in \{0,1\}^* \mid ceros(w) = unos(w), w = z1\}$$

Siguiendo el mismo razonamiento se puede deducir que:

$$\textsc{BinIg} - \textsc{BinImpar} = \{w \in \{0,1\}^* \mid (ceros(w) = unos(w) \wedge w = z0) \vee w = \lambda\}$$

$$\overline{\textsc{BinIg}} = \{0,1\}^* - \textsc{BinIg} = \{w \in \{0,1\}^* \mid ceros(w) \neq unos(w)\}$$

Propiedades básicas de operaciones de lenguajes como conjuntos

Dados tres lenguajes cualesquiera L_1, L_2, L_3, incluidos en cierto lenguaje universal V^* (respecto del cual se hace la complementación), se cumplen las siguientes propiedades:

- **Asociativa**: $L_1 \cap (L_2 \cap L_3) = (L_1 \cap L_2) \cap L_3$ y $L_1 \cup (L_2 \cup L_3) = (L_1 \cup L_2) \cup L_3$
- **Conmutativa**: $L_1 \cap L_2 = L_2 \cap L_1$ y $L_1 \cup L_2 = L_2 \cup L_1$
- **Distributiva**:
  1. De la intersección respecto de unión: $L_1 \cap (L_2 \cup L_3) = (L_1 \cap L_2) \cup (L_1 \cap L_3)$
  2. De la unión respecto de intersección: $L_1 \cup (L_2 \cap L_3) = (L_1 \cup L_2) \cap (L_1 \cup L_3)$
- **Leyes de De Morgan**: $\overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$ y $\overline{L_1 \cup L_2} = \overline{L_1} \cap \overline{L_2}$
- **Propiedades del complementario**: $\overline{L_1} \cap L_1 = \varnothing$, $\overline{L_1} \cup L_1 = V^*$ y $\overline{\overline{L_1}} = L_1$
- **Propiedades del lenguaje vacío**: $L_1 \cap \varnothing = \varnothing$ y $L_1 \cup \varnothing = L_1$
- **Propiedades del lenguaje universal**: $L_1 \cap V^* = L_1$ y $L_1 \cup V^* = V^*$

4.2. Concatenación, potencia y reflexión de lenguajes

Las operaciones definidas entre cadenas se pueden *extender a lenguajes* de la siguiente forma:

Definición 6 *Sean L_1 y L_2 lenguajes:*

- *La **concatenación** de L_1 y L_2, denotada como $L_1 \circ L_2$ es el lenguaje definido como:*

$$\boxed{L_1 \circ L_2 = \{x \circ y \mid x \in L_1, y \in L_2\} = \{xy \mid x \in L_1, y \in L_2\}}$$

  Ej. Si $L_1 = \{ab, c, \lambda\}$ y $L_2 = \{cc, a\}$ entonces:
  $\{ab, c, \lambda\} \circ \{cc, a\} = \{ab \circ cc, ab \circ a, c \circ cc, c \circ a, \lambda \circ cc, \lambda \circ a\} = \{abcc, aba, ccc, ca, cc, a\}$.

- *La **potencia n-ésima** de un lenguaje L_1, denotada L_1^n, es el lenguaje que resulta de concatenar el lenguaje L_1 consigo mismo n veces. Formalmente se define de forma recursiva como:*

  1. *Regla base: para $n = 0$ se tiene que* $\boxed{L_1^0 = \{\lambda\}}$
  2. *Regla recursiva: si $n > 0$ entonces* $\boxed{L_1^n = L_1 \circ L_1^{n-1}}$

  Ej. Dado el lenguaje finito $A = \{ab, c\}$ con alfabeto $V = \{a, b, c\}$:

  $A^0 = \{\lambda\}$
  $A^1 = A \circ A^0 = \{ab, c\} \circ \{\lambda\} = \{ab \circ \lambda, c \circ \lambda\} = \{ab, c\}$
  $A^2 = A \circ A^1 = \{ab, c\} \circ \{ab, c\} = \{ab \circ ab, ab \circ c, c \circ ab, c \circ c\} = \{abab, abc, cab, cc\}$
  $A^3 = A \circ A^2 = \{ab, c\} \circ \{abab, abc, cab, cc\} =$
  $\{ababab, ababc, abcab, abcc, cabab, cabc, ccab, ccc\}$

  Obsérvese que no se puede afirmar que $L^n = \{w^n \mid w \in L\}$, ya que para $n = 2$ y $L = A = \{ab, c\}$ se tiene que $\{w^2 \mid w \in \{ab, c\}\} = \{(ab)^2, c^2\} = \{abab, cc\}$ y este lenguaje es claramente distinto de $A^2 = \{abab, abc, cab, cc\}$.

- *El **lenguaje reflejo** de L_1, denotado L_1^R, es el lenguaje que contiene todas las cadenas reflejas de las cadenas de L_1:*

$$L_1^R = \left\{w^R \mid w \in L_1\right\}$$

  Ej. El lenguaje reflejo de $A = \{ab, c\}$ es el lenguaje $A^R = \{ab, c\}^R = \left\{(ab)^R, c^R\right\} = \{ba, c\}$.

PROPIEDADES DE LA CONCATENACIÓN DE LENGUAJES:

Dados tres lenguajes cualesquiera L_1, L_2, L_3, se cumplen las siguientes propiedades:

- **Elemento nulo $\varnothing$:** $L_1 \circ \varnothing = \varnothing$

- **Elemento identidad $\{\lambda\}$:** $L_1 \circ \{\lambda\} = \{\lambda\} \circ L_1 = L_1$
- **Propiedad asociativa:** $(L_1 \circ L_2) \circ L_3 = L_1 \circ (L_2 \circ L_3)$
- **Distributiva de la concatenación de lenguajes con respecto de la unión:**
  1. $L_1 \circ (L_2 \cup L_3) = (L_1 \circ L_2) \cup (L_1 \circ L_3)$
  2. $(L_2 \cup L_3) \circ L_1 = (L_2 \circ L_1) \cup (L_3 \circ L_1)$

4.3. Clausura de Kleene de un lenguaje

Dado un lenguaje L, se define la **clausura de Kleene** (o *cierre*) de L, denotado L^*, mediante cualquiera de las dos expresiones siguientes:

$$\overset{(1)}{\boxed{L^* = \bigcup_{n=0}^{\infty} L^n}} \quad \text{o por comprensión:} \quad \overset{(2)}{\boxed{L^* = \{w_1 w_2 \dots w_k \mid k \geq 0, \text{ donde cada } w_i \in L\}}}$$

Al desarrollar la unión infinita en (1) se tiene que $L^* = L^0 \cup L^1 \cup L^2 \cup \dots$ y puede expresarse de forma explícita como se indica en la definición (2), donde se aprecia que L^* contiene todas las cadenas que se obtienen al concatenar cero o más cadenas de L; se tiene que $\lambda \in L^*$ por defecto, cuando se considera $k = 0$ (se concatenan 0 cadenas de L). De la expresión (1) se deduce igualmente que $\boxed{\lambda \in L^*}$ sea cual sea el lenguaje L, porque al hacer la unión con $L^0 = \{\lambda\}$ se tiene que $L^* = \{\lambda\} \cup L^1 \cup L^2 \cup \dots$.

Se define la **clausura positiva** (o *cierre positivo*) de L, denotado L^+, de forma análoga a L^*, pero considerando que se concatenan una o más cadenas de L:

$$\boxed{L^+ = \bigcup_{n=1}^{\infty} L^n} \text{ o por comprensión: } \boxed{L^+ = \{w_1 w_2 \dots w_k \mid k > 0, \text{ donde cada } w_i \in L\}}$$

A partir de las definiciones anteriores se deduce que $\boxed{L^+ \subseteq L^*}$ puesto que $\boxed{L^* = \{\lambda\} \cup L^+}$ Aunque en la unión infinita de L^+ no se incluye la potencia cero $L^0 = \{\lambda\}$, eso no quiere decir necesariamente que $\lambda \notin L^+$ porque puede que $\lambda \in L$ y por tanto se incluye en la clausura positiva por la unión con $L^1 = L$, o bien, por la definición explícita: si $\lambda \in L$ entonces para $k = 1$ y $w_1 = \lambda$ se tiene que $\lambda \in L^+$.

Ejemplo 14 *Sea el lenguaje $C = \{00, 11\}$. En C^* tendríamos todas las cadenas que se forman al concatenar cero o más veces las cadenas 00 o 11. Ejemplos de cadenas que pertenecen a C^* son (de menor a mayor longitud, hasta longitud 4):*

$$\lambda, 00, 11, 0000, 0011, 1100, 1111, \dots$$

Para describir C^* de forma explícita por comprensión recurrimos a la definición general dada en (2) más arriba, sustituyendo L por $\{00, 11\}$:

$$C^* = \{w_1 w_2 \dots w_k \mid k \geq 0, \text{ donde cada } w_i \in \{00, 11\}\}$$

lo cual equivale a:

$$C^* = \{w_1 w_2 \dots w_k \mid k \geq 0, \text{ donde cada } w_i = 00 \vee w_i = 11\}$$

Sean ahora los lenguajes $A = \{a\}$ *y* $B = \{b\}$*. Vamos a describir por comprensión* $(A \circ B)^*$ *y* $(A \circ B)^+$ *con una expresión lo más corta posible, usando un patrón con parámetros numérico. Primero calculamos* $A \circ B = \{ab\}$*, que contiene sólo la cadena* ab *y por tanto en la clausura* $(A \circ B)^*$ *tendríamos todas las cadenas que se forman al concatenar cero o más veces la cadena* ab *consigo misma. Por tanto podemos afirmar que:*

$$(A \circ B)^* = \{ab\}^* = \{(ab)^n \mid n \geq 0\}$$

Desarrollando la unión de las sucesivas potencias de $(A \circ B)^*$ *podemos llegar a la misma conclusión:*

$$(AB)^* = \textstyle\bigcup_{n=0}^{\infty}(AB)^n = (AB)^0 \cup (AB)^1 \cup (AB)^2 \cup (AB)^3 \cup \dots =$$
$$\{\lambda\} \cup \{ab\} \cup \{abab\} \cup \{ababab\} \cup \dots = \{(ab)^n \mid n \geq 0\}$$

Por razonamiento análogo se tiene que $(A \circ B)^+ = \{ab\}^+ = \{(ab)^n \mid n > 0\}$*. En este caso* $\lambda \notin (A \circ B)^+$ *porque* $\lambda \notin (A \circ B)$*.*

El **lenguaje universal** V^* se definió informalmente como el conjunto que contiene a todas las cadenas formadas por símbolos de un determinado alfabeto V y la cadena vacía. Ahora podemos definirlo también como la clausura del alfabeto V, esto es:

$$\boxed{V^* = \bigcup_{n=0}^{\infty} V^n}$$

Con esta expresión podemos comprobar que cualquier cadena del lenguaje universal puede generarse por concatenación de cero o más símbolos de V, por eso se dice que *el alfabeto* V *es un generador* del lenguaje universal V^*.

5. Preguntas de evaluación

Aparte de los **14 ejemplos** de los apuntes, en esta sección tenemos **preguntas que sirven como auto-evaluación de los conocimientos teóricos**. Abarcan problemas de razonamiento o demostración, problemas de aplicación de definiciones y preguntas tipo test. Muchas de estas preguntas han aparecido en exámenes de cursos anteriores, aunque eso no significa que las preguntas que puedan aparecer en un examen sean exactamente del tipo de las que se incluyen aquí.

Nota:

- En los problemas donde se pide una **demostración o justificación** hay que dar un argumento formal y completo, usando conceptos o propiedades teóricas y mostrando todos los pasos necesarios para deducir lo que se quiere probar. En nivel de detalle puede ser similar a las argumentaciones que aparecen en los problemas resueltos.
- Los problemas de **mayor dificultad** se señalan con **(!)**.

5.1. Problemas resueltos

1. Demuestra si se cumple la propiedad distributiva de la concatenación de lenguajes respecto de la intersección, es decir, si para tres lenguajes A, B, C cualesquiera se cumple lo siguiente:

$$\overbrace{A \circ (B \cap C)}^{(1)} = \overbrace{(A \circ B) \cap (A \circ C)}^{(2)}$$

Solución. No se cumple la propiedad y para ello hacemos una **demostración por contraejemplo.** Escogemos tres lenguajes concretos: $A = \{a, \lambda\},\ B = \{\lambda\},\ C = \{a\}$. Calculamos el resultado de las operaciones en las expresiones (1) y (2) de ambos miembros de la igualdad.

$$\begin{array}{ll} (1): & A \circ (B \cap C) = \{a, \lambda\} \circ (\{\lambda\} \cap \{a\}) = \varnothing \\ (2): & (A \circ B) \cap (A \circ C) = \{a, \lambda\} \cap \{aa, a\} = \{a\} \end{array}$$

Como vemos, se obtienen resultados diferentes para (1) y (2), por tanto no se cumple la propiedad.

2. **(!)** Sea el alfabeto $V = \{0, 1\}$ y los lenguajes:

$$\begin{array}{l} L_1 = \{w \in \{0,1\}^* \mid ceros(w) \text{ es par}\} \\ L_2 = \{w \in \{0,1\}^* \mid w = 01^n, n \geq 0\} \end{array}$$

Demuestra que $L_1 \circ L_2 = L_3$, donde:

$$L_3 = \{w \in \{0,1\}^* \mid ceros(w) \text{ es impar}\}$$

Solución. Para demostrar que $L_1 \circ L_2 = L_3$ se procede como en una **demostración de igualdad de conjuntos**, por lo que hay que probar que se da la inclusión en ambos sentidos, es decir, que $L_1 \circ L_2 \subseteq L_3$ y que $L_3 \subseteq L_1 \circ L_2$.

Probamos que $L_1 \circ L_2 \subseteq L_3$. Mostramos los pasos para deducirlo.
1. Si $w \in L_1$ entonces $ceros(w)$ es par, por definición de L_1.
2. Si $w' \in L_2$ entonces w' sólo tiene un cero, por definición de L_2.
De 1. y 2. se deduce que $ww' \in L_1 \circ L_2$ es una cadena que tiene en total un número impar de ceros, por lo que pertenece a L_3. Como w, w' son cadenas arbitrarias de L_1 y L_2, respectivamente, entonces se cumple que $L_1 \circ L_2 \subseteq L_3$, como queríamos probar.

Probamos que $L_3 \subseteq L_1 \circ L_2$. Tenemos que probar que cada cadena w con un número impar de 0's (pertenece a L_3) puede obtenerse como concatenación de una cadena de L_1 seguida de una cadena de L_2. Hacemos una **demostración por casos**, comprobando esta condición en el caso de que la cadena termine en 0 y en el caso de que termine en 1 (cubre todos los casos de cadenas de L_3).

a) Supongamos que $w \in L_3$ y $w = x0$. Entonces x debe tener un número par de ceros, por tanto:

$$w = \underbrace{x}_{\in L_1}\underbrace{0}_{\in L_2}$$

b) Supongamos que $w \in L_3$ y $w = x1$. Entonces x debe tener un número impar de ceros, por pertenecer w a L_3. Ahora partimos la cadena x de forma $x = z_1 0 z_2$, donde el cero que aparece es el de más a la derecha de x y por tanto z_1 será una subcadena con un número par de ceros y z_2 una subcadena con cero o más unos. Por tanto.

$$w = x1 = \underbrace{z_1}_{\in L_1}\underbrace{0z_2 1}_{\in L_2}$$

De los casos *a*) y *b*) se deduce que toda cadena w del lenguaje L_3 puede expresarse como la concatenación de una cadena de L_1 y una cadena de L_2, por tanto $L_3 \subseteq L_1 \circ L_2$, como queríamos probar.

CONCLUSIÓN: habiendo probado que $L_1 \circ L_2 \subseteq L_3$ y que $L_3 \subseteq L_1 \circ L_2$ se deduce que $\boxed{L_1 \circ L_2 = L_3}$

3. Describir el lenguaje N_{par} formado por cadenas que representan números naturales pares (pueden comenzar por 0) por comprensión, sin condiciones numéricas.

   Solución. El alfabeto de este lenguaje es $V_{dig} = \{0, 1, \ldots, 9\}$. Entonces:

   $$N_{par} = \{xp \in | \ x \in V^*_{dig}, p \in \{0, 2, 4, 6, 8\}\}$$

   La descripción muestra un patrón para las cadenas de números pares. Dice en el preámbulo que están formadas por la concatenación de dos subcadenas (xp) y en el cuerpo se indica que x es una secuencia de cero o más dígitos ($x \in V^*_{dig}$) y se impone la condición de que p sea un dígito par ($p \in \{0, 2, 4, 6, 8\}$).
   Ej. se puede afirmar que $334 \in N_{par}$ según la definición anterior porque $334 = xp$, donde $x = 33$, que pertenece a V^*_{dig}, y $p = 4$, que pertenece al conjunto $\{0, 2, 4, 6, 8\}$. También tenemos que $6 \in N_{par}$ porque $6 = xp$, donde $x = \lambda$, $\lambda \in V^*_{dig}$, $p = 6$ y $6 \in \{0, 2, 4, 6, 8\}$. Sin embargo $845 \notin N_{par}$, porque debe ser $p = 5$, pero $5 \notin \{0, 2, 4, 6, 8\}$.

4. Consideremos de nuevo el lenguaje N_{par}. Demuestra la verdad o falsedad de las siguientes afirmaciones.

   a) $N_{par} = N^+_{par}$

   b) $N_{par} = N^*_{par}$

   Solución.
   a) Primero observamos que para formar la clausura positiva N^+_{par} se obtienen todas las cadenas resultantes de la concatenación de una o más cadenas de N_{par} y de esta forma se obtienen siempre cadenas que acaban con un dígito par, luego todas las cadenas $w \in N^+_{par}$ cumplen que $w \in N_{par}$, lo que significa que $N^+_{par} \subseteq N_{par}$. Por otro lado esta claro que $N_{par} \subseteq N^+_{par}$ puesto que por definición $N^+_{par} = \bigcup_{n=1}^{\infty} N^n_{par}$. Como $N^+_{par} \subseteq N_{par}$ y $N_{par} \subseteq N^+_{par}$ podemos concluir que $\boxed{N_{par} = N^+_{par}}$.

   b) $\boxed{N_{par} \neq N^*_{par}}$ porque $\lambda \notin N_{par}$ (no acaba en dígito par) y $\lambda \in N^*_{par}$ (la cadena vacía pertenece a la clausura de cualquier lenguaje).

5. Sea el lenguaje $A = \{\lambda, a\}$.

a) Obtener A^n para $0 \leq n \leq 3$, teniendo en cuenta la definición recursiva de A^n.

b) ¿Cuantos elementos tiene A^n en general ($\forall\, n \geq 0$)? Justifica la respuesta

c) ¿Se cumple que $A^* = A^+$? Justifica la respuesta.

Solución.

a) Calculamos las potencias de A hasta $n = 3$, según la definición recursiva:

$$\begin{array}{l} A^0 = \{\lambda\} \text{ (caso base)} \\ A^1 = A \circ A^0 = \{\lambda, a\} \circ \{\lambda\} = \{\lambda, a\} \\ A^2 = A \circ A^1 = \{\lambda, a\} \circ \{\lambda, a\} = \{\lambda, a, aa\} \\ A^3 = A \circ A^2 = \{\lambda, a\} \circ \{\lambda, a, aa\} = \{\lambda, a, aa, aaa\} \end{array}$$

b) Observando las potencias anteriores y generalizando podemos deducir que:

$$A^n = \{a^k \mid 0 \leq k \leq n\}$$

Para $k = 0$ se tiene que $a^0 = \lambda \in A^n$. Por tanto, el cardinal de A^n es:

$$\boxed{|A^n| = n + 1, \forall\, n \geq 0}$$

c) Teniendo en cuenta que $A^* = \bigcup_{n=0}^{\infty} A^n = A^0 \cup A^1 \cup A^2 \cup \ldots$ (1) y teniendo en cuenta las cadenas que hay en cada potencia A^n (como se muestra para $n = 0, 1, 2, 3$ y se generaliza más arriba para cualquier n) y podemos deducir que:

$$A^* = \bigcup_{n=0}^{\infty} A^n = \{a^n \mid n \geq 0\}$$

Por otra parte $A^+ = \bigcup_{n=1}^{\infty} A^n = A^1 \cup A^2 \cup \ldots$ (2). Pero como $\lambda \in A^1 = A$ (en realidad λ pertenece a cualquier potencia de A en el desarrollo de A^+) entonces no hay diferencia entre (1) y (2), luego $\boxed{A^* = A^+}$

5.2. Problemas propuestos

1. Sea el lenguaje L_1 que se describe informalmente como: "cadenas que empiezan por una o más $a's$, seguidas de una c y acaban en una secuencia opcional de $b's$ (cero o más $b's$).

   a) Describe el alfabeto de L_1 y el lenguaje universal.

   b) Describe L_1 por comprensión.

   c) Obtén todas las cadenas w que cumplen: $w \in L_1$ y $|w| \leq 3$

   d) Justifica si las siguientes cadenas pertenecen o no a L_1^* o a L_1^+:

   $$\lambda, cbb, ac, acacb, aacbcbb, acbacbbac$$

   e) Sea $L_2 = \{cb, aa, acbc\}$. Justifica si se cumplen o no las siguientes afirmaciones:

[1] $\exists w, z : w \in L_1 \wedge z \in L_2 \wedge z$ es prefijo de w.

[2] $\exists w, z : w \in L_1 \wedge z \in L_2 \wedge z$ es sufijo de w.

f) Describe el lenguaje $L_1^R \circ L_1$ por comprensión.

g) Obtén todas las cadenas w que cumplen: $w \in L_1^R \circ L_1$ y $|w| \leq 5$.

2. Se dice que una cadena w es un palíndromo si se lee igual de izquierda a derecha que de derecha a izquierda. Indica una definición por comprensión lo más breve posible del lenguaje de palíndromos con alfabeto $V = \{a, b\}$, que llamamos $PALIN_{ab}$.

3. Sean los lenguajes $A = \{ab^n a \mid n \geq 0\}$ y $B = \{a^i b^j a^i \mid 0 \leq i < j\}$. Describe formalmente los siguientes lenguajes, usando patrones con parámetros numéricos para describir las cadenas, en caso de que el resultado sea un lenguaje infinito.

   a) $A \cap B$ b) $A - B$

   Indica cuál es la cadena (o cadenas en caso de haber varias) de menor longitud que pertenecen a cada lenguaje.

4. Consideremos el lenguaje finito $D_{par} = \{0, 2, 4, 6, 8\}$ de las cadenas correspondientes a dígitos decimales pares. Demuestra si se cumple la igualdad $D_{par}^+ = N_{par}$, donde N_{par} es el lenguaje de los números naturales pares.

5.3. Preguntas tipo test

Para cada pregunta señala la opción correcta. Como ejercicio adicional, intenta razonar la verdad o falsedad de cada opción. Una pregunta tipo test también puede enunciarse en un examen como pregunta de justificar la verdad o falsedad de una afirmación.

1. Señala la opción <u>verdadera</u> (los errores de notación hacen que la opción sea falsa). Dado un alfabeto V,

   a) $\{\lambda\} \in V^*$

   b) $\lambda \in V$

   c) $\{\lambda\} \subseteq V^* \wedge \exists\, L \subseteq V^* : \lambda \in L$

2. Señala la opción <u>falsa</u>. Dadas $x, y \in V^*$,

   a) $|(xy)^n| = n \times (|x| + |y|)$

   b) $(xy)^R = x^R y^R$

   c) $|x^i y^j| = i \times |x| + j \times |y|$

3. Señala la opción <u>verdadera</u>. Sea L un lenguaje con alfabeto V,

   a) $V^* \cap L = L$

   b) $L \circ \varnothing = L$

   c) $L^+ = L^* - \{\lambda\}$

4. (!) Señala la opción <u>verdadera</u>. Dados los lenguajes L_1, L_2, L_3,

   a) $(L_1 \circ L_2)^R \circ L_3 = L_2^R \circ L_1^R \circ L_3$

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

   c) $(L_1 \circ L_2)^R = L_1^R \circ L_2^R$

5. Señala la opción <u>falsa</u>. Dado el lenguaje $A = \{ww \mid w \in \{a, b\}^*\}$,

   a) $z \in A^* \Rightarrow |z|$ es par

   b) $A^+ = A$

   c) $z \in A \Rightarrow z^R \in A$