Universidad Nacional Autónoma de México
Facultad de Ciencias
Teoría de la Computación
Notas de Clase
Profesor: Enrique F. Soto-Astorga, MA/MSc.
Ayudante: Laura Itzel Rodríguez Dimayuga

Índice

1. Historia e Introducción

1.1. Turing 1936

1.2. Hilbert y el Entscheidungsproblem

1.3. Gödel y la incompletitud

1.4. Tesis de Church-Turing-Post

1.5. Automatas, Computabilidad y Complejidad

1.6. Incomputabilidad

2. Cadenas y alfabetos

Definición 1

Un *alfabeto* es un conjunto finito no vació. Para referirnos al alfabeto utilizaremos Σ

Ejemplo:

- $\Sigma = \{0, 1\}$ es un alfabeto binario.
- $\Sigma = \{a, b, c, \ldots z\}$ es un alfabeto de letras.
- $\Sigma =$ {El conjunto de todos los símbolos ASCII}

Definición 2

Los *símbolos* o *letras* son elementos de Σ, es decir, de nuestro alfabeto.

Ejemplo:

- Si $\Sigma = \{0, 1\}$, entonces 0 y 1 son símbolos de Σ.
- Si $\Sigma = \{a, b, c, \ldots z\}$, entonces a, b, c, ..., z son símbolos de Σ.
- Si $\Sigma =$ {El conjunto de todos los símbolos ASCII}, entonces cada símbolo ASCII es un símbolo de Σ.

Definición 3

Una *cadena* (string) es una secuencia finita de elementos de Σ.

Ejemplo:

- Si $\Sigma = \{0, 1\}$, entonces 0011 es una cadena sobre Σ.
- Si $\Sigma = \{a, b, c, \ldots z\}$, entonces abc es una cadena sobre Σ.
- Si $\Sigma =$ {El conjunto de todos los símbolos ASCII}, entonces $Hello\$\#@$ es una cadena sobre Σ.

Definición 4

El *tamaño* de una cadena es la cantidad de símbolos que la componen.

Ejemplo:

- Si $\Sigma = \{0, 1\}$, entonces $|0011| = 4$.

- Si $\Sigma = \{a, b, c, \ldots z\}$, entonces $|abc| = 3$.
- Si $\Sigma = \{\text{El conjunto de todos los símbolos ASCII}\}$, entonces $|Hello\$\#@| = 8$.

Definición 5

A la *cadena vacía* la llamamos ϵ o λ, de manera de que $|\lambda| = 0$

La cadena vacía es diferente al conjunto vacío $\lambda \neq \varnothing$. Más adelante veremos que siempre aparece en conjunto de todas las cadenas (Σ^*) sobre un alfabeto.

Definición 6

La *reversa* de una cadena w se denota como w^R y se obtiene invirtiendo el orden de los símbolos en w.

Ejemplo:
Si $w = 0011$, entonces $w^R = 1100$.Podemos definir a al conjunto de las palabras palindromas[1] como:

$$P = \{w \in \Sigma^* \mid w = w^R\}$$

Definición 7

Decimos que x es *prefijo* de y, si existe una cadena z tal que $y = xz$. Análogamente decimos que x es *sufijo* de y si existe una cadena z tal que $y = zx$.

- La cadena vacía es prefijo y sufijo de cualquier cadena.

Ejemplos:

- Si $w = 0011$ y $x = 00$, entonces x es prefijo de w, ya que existe $z = 11$ tal que $0011 = 00 \cdot 11$.
- Si $w = 0011$ y $y = 11$, entonces y es sufijo de w, ya que existe $z = 00$ tal que $0011 = 00 \cdot 11$.

3. Operaciones sobre alfabetos, lenguajes y cadenas

Definición 8

La *concatenación* de dos cadenas w e z se denota como $w \cdot z$ ó wz y es la cadena que resulta de unir w e z.
Nota: La concatenación aplica para **símbolos**, **cadenas** y **lenguajes**.

Ejemplo:

- Si $\Sigma = \{0, 1\}$, entonces 01 es una concatenación sobre **símbolos** de Σ.
- Si $w = 01$ e $z = 10$, entonces $wz = w \cdot z = 0110$ es una concatenación sobre **cadenas**.
- Si $L_1 = \{0, 1\}$ y $L_2 = \{1, 0\}$, entonces $L_1L_2 = \{01, 00, 10, 11\}$ es una concatenación sobre **lenguajes**.

[1] Una palabra es palíndroma si se lee igual de izquierda a derecha que de derecha a izquierda.

Definición 9

Podemos definir la *potencia* de una cadena w como $w^n = w \cdot w \cdots w$ (n veces), donde n es un número natural. Por ejemplo, si $w = 01$, entonces $w^3 = 010101$.
De manera inductiva tenemos que:

- $w^0 = \lambda$ (la cadena vacía)
- $w^{n+1} = w^n \cdot w$

La potencia la podemos aplicar a **símbolos**, **cadenas**,**lenguajes** y **alfabetos**.

Ejemplo:
Si w es una **cadena**, digamos $w = 01$, entonces:

- $w^0 = \lambda$
- $w^1 = 01$
- $w^2 = 0101$
- $w^3 = 010101$

Si L es un **lenguaje**, digamos $L = \{00, 1\}$, entonces:

- $L^0 = \{\lambda\}$
- $L^1 = \{00, 1\}$
- $L^2 = \{(00)00, (00)1, 1(00), (1)1\} = \{0000, 001, 100, 11\}$

Para lenguajes L_1 y L_2, la potencia se define como:

- $L_1^0 = \{\lambda\}$
- $L_1^1 = L_1$
- $L_1^2 = \{xy \mid x \in L_1, y \in L_1\}$
- $L_1^3 = \{xyz \mid x \in L_1, y \in L_1, z \in L_1\}$

Aquí hay otro ejemplo con $L = \{ab, aab\}$:

$\{ab, aab\}^0 = \{\epsilon\}$
$\{ab, aab\}^1 = \{ab, aab\}$
$\{ab, aab\}^2 = \{abab, abaab, aabab, aabaaab\}$
$\{ab, aab\}^3 = \{ab(ab)ab, ab(ab)aab, ab(aab)ab, (aab)abab, aab(aab)ab, aab(ab)aab, ab(aab)aab, aab(aab)aab\}$

Como tengo 2 cadenas y quiero ver todas las combinaciones de 2 cadenas, entonces tengo $2^2 = 4$ combinaciones. Si quiero ver todas las combinaciones de 3 cadenas, entonces tengo $2^3 = 8$ combinaciones.

Definición 10

También podemos definir la *potencia* de un alfabeto Σ como $\Sigma^n = \{w^n \mid w \in \Sigma^*\}$, donde Σ^* es el conjunto de todas las cadenas sobre Σ.

Por ejemplo, si $\Sigma = \{0, 1\}$, entonces $\Sigma^2 = \{00, 01, 10, 11\}$.

$$
\begin{aligned}
\Sigma^0 &= \{\lambda\} \\
\Sigma^1 &= \{0, 1\} \\
\Sigma^2 &= \{00, 01, 10, 11\} \\
\Sigma^3 &= \{000, 001, 010, 011, 100, 101, 110, 111\} \\
&\vdots \\
\Sigma^n &= \{w^n \mid w \in \Sigma^*\}
\end{aligned}
$$

Definición 11

Vamos a definir a la *cerradura de Kleene* o la *estrella de Kleene* de un alfabeto Σ como Σ^*,

$$
\begin{aligned}
\Sigma^* &= \bigcup_{n \geq 0} \Sigma^n = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots \\
\Sigma^+ &= \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots \\
\Sigma^* &= \Sigma^+ \cup \{\lambda\}
\end{aligned}
$$

Donde Σ^n es la potencia del alfabeto. Podemos describir a Σ^*(sigma estrella) como el conjunto de todas las cadenas (incluida la cadena vacía) que se pueden formar a partir de los símbolos de Σ.

Ejemplo:
Si $\Sigma = \{0, 1\}$, entonces

$$\Sigma^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \ldots\}$$

es decir, el conjunto de todos los números binarios (incluidos los de longitud cero, a.k.a. la cadena vacía).

Lema 1

Sea Q un conjunto finito. Entonces, el número de subconjuntos de Q es $2^{|Q|}$, donde $|Q|$ es el número de elementos en Q. A este conjunto se le llama *conjunto potencia* de Q y se denota como $\mathcal{P}(Q)$ ó 2^Q.

Demostración. Sea Q un conjunto finito con n elementos. Entonces, cada elemento de Q puede estar presente o no en un subconjunto, lo que nos da dos opciones (incluir o no incluir) cada elemento. Por lo tanto, el número total de subconjuntos de Q es $2^n = 2^{|Q|}$. ■

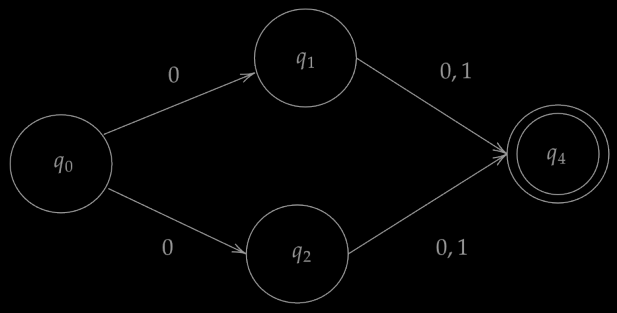

Figura 1: Ejemplo de un autómata finito determinista (DFA). Este es el autómata que acepta la expresión regular $0(0+1) = \{0, 00, 01\}$.

Más adelante cuando veamos autómatas finitos no deterministas, veremos su función de transición, la cual es una función que toma un estado y un símbolo de entrada y devuelve un conjunto de estados. Esta función se denota como $\delta : Q \times \{\Sigma \cup \lambda\} \to 2^Q$ (conjunto potencia), donde Q es el conjunto de estados y Σ es el alfabeto.

3.1. Operaciones sobre cadenas

Definición 12

- **Concatenación:** Dadas dos cadenas x e y, la concatenación es asociativa: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$ para toda cadena.
- La cadena vacía λ es la **identidad** de la concatenación: $x \cdot \lambda = \lambda \cdot x = x$ para toda cadena x.
- $|x \cdot y| = |x| + |y|$ para toda cadena x e y.
- $w^m w^n = w^{m+n}$

El conjunto de $(\Sigma^*, \cdot)$ con λ como identidad es un monoide, ya que cumple con las propiedades de asociatividad e identidad.

3.2. Lenguajes y operaciones sobre lenguajes

Definición 13

Dado un alfabeto Σ, un **lenguaje** sobre Σ es un subconjunto de Σ^*. Dado un lenguaje $L \subseteq \Sigma^*$, definimos las siguientes operaciones sobre lenguajes:

- **Unión:** $L(a) \cup L(b) = L(a+b) = \{w \mid w \in L_1 \text{ o } w \in L_2\}$
- **Intersección:** $L_1 \cap L_2 = \{w \mid w \in L_1 \text{ y } w \in L_2\}$
- **Concatenación:** $L(a) \cdot L(b) = L(ab) = \{wz \mid w \in L(a), z \in L(b)\}$
- **Diferencia:** $L_1 - L_2 = \{w \mid w \in L_1 \text{ y } w \notin L_2\}$
- **Complemento:** $\overline{L} =\sim L = \{\lambda\} \cup (\Sigma^* - L)$
- **Cerradura de Kleene:** $L^* = \bigcup_{n \geq 0} L^n$
- $L^+ = L^* - \{\lambda\} = \bigcup_{n \geq 1} L^n$

Una analogía útil es pensar en los lenguajes es como el el lenguaje español. Sabemos que nuestro alfabeto es $\Sigma = \{\texttt{a, b, c, ..., z}\}$ y que podemos formar palabras (cadenas) con estos símbolos.El lenguaje español $L(esp)$ es un subconjunto de Σ^*. Otro ejemplo es pensar en un lenguaje de programación como C, donde el alfabeto es el conjunto de caracteres ASCII y el lenguaje es el conjunto de todas las cadenas que son programas válidos en ese lenguaje.

Algunos ejemplos de lenguajes son:

1. El lenguaje de todas las cadenas que contienen tienen el mismo número de 0s y 1s. Para contar el número de caracteres dentro de una cadena podemos usar la notación $\#0(w)$ para el número de 0s y $\#1(w)$ para el número de 1s. Entonces, este lenguaje se puede definir como:

$$L = \{w \in \Sigma^* \mid \#0(w) = \#1(w)\}$$

2. El lenguaje de todas las cadenas que comienzan con 0 y terminan con 1. Este lenguaje se puede definir como:

$$L = \{w \in \Sigma^* \mid w = 0x1, x \in \Sigma^*\}$$

3. El lenguaje de todas los números binarios que son primos

0 = 0	No es primo
1 = 1	No es primo
10 = 2	Es primo
11 = 3	Es primo
100 = 4	No es primo
101 = 5	Es primo
110 = 6	No es primo
111 = 7	Es primo
1000 = 8	No es primo
1001 = 9	No es primo
1010 = 10	No es primo
1011 = 11	Es primo
1100 = 12	No es primo
1101 = 13	Es primo
1110 = 14	No es primo
1111 = 15	No es primo

$$\{10, 11, 101, 111, 1011, 1101, \dots\}$$

Seguramente alguna vez hemos programado un verificador de números primos en algún lenguaje de programación. En general, veremos que los lenguajes son conjuntos de cadenas que cumplen ciertas propiedades. Algunos lenguajes son capturados por expresiones regulares o autómatas.

3.2.1. Muchas cosas se pueden resolver con lenguajes y autómatas, y muchas cosas que no

Una de las cosas que resuelven los autómatas es decidir si una cadena pertenece o no a un lenguaje. Este es un problema de decisión (Entscheidungsproblem), gracias a Turing sabemos que no podemos obtener un sí o no para cualquier problema.

Un *problema de decisión* puede ser determinar si una cadena w pertenece a un lenguaje L. Por ejemplo, pensemos en el lenguaje de los números primos, si tenemos cualquier cadena podríamos preguntarnos si esta representación binaria es un número primo. ¿Dada una cadena w, $w \in L_p$?.

En el siguiente capítulo trataremos las cadenas que sabemos que un autómata sí puede responder. Veremos que este autómata tiene limitaciones y más adelante trataremos de rebasarlas con el autómata con pila. Por ejemplo, el lenguaje $\{0^n 1^n \mid n \geq 0\}$ es el conjunto de todas las cadenas que tienen el mismo número de 0's seguido del mismo número de 1's. Este lenguaje no puede ser reconocido por un autómata finito, pero si por un autómata con pila. Aunque este último tipo de autómata tiene más potencia, todavía hay lenguajes que no puede reconocer. Y más adelante veremos que hay lenguajes que no pueden ser reconocidos por ningún autómata, ni siquiera por una máquina de Turing. El quiebre del programa de Hilbert por querer encontrar un procedimiento efectivo para decidir si una proposición es verdadera o falsa quedó muerto.

3.2.2. Operaciones sobre lenguajes

Ejemplo 1. Sea $(L \in \Sigma^* = \{0, 1\}^*)$.

- El lenguaje de todas las cadenas que empiezan con 0 es $L_0 = \{0w \mid w \in \Sigma^*\} = \{0, 01, 00, 000, 001, \cdots\}$.

Figura 2: David Hilbert (1862-1943), un matemático alemán triste.

- El lenguaje de todas las cadenas que terminan con 1 es $L_1 = \{w1 \mid w \in \Sigma^*\} = \{1, 01, 11, 001, 011, \cdots\}$
- **Unión:** $L_1 \cup L_2 = \{0, 1, 00, 01, 11, 000, 001, 010, 011, \cdots\} = \{0w \ \cup \ w1 \mid w \in \Sigma^*\}$
- **Intersección:** $L_1 \cap L_2 = \{01, 001, 011, 0001, \cdots\}$
- **Concatenación:** $L_1L_2 = \{01, 001, 011, 101, 0101, \cdots\}$ (La concatenación es como un producto cruz de los elementos de ambos lenguajes concatenados)
- **Diferencia:** $L_1 - L_2 = \{0, 00, 000, 001, \cdots\}$
- **Complemento:** $\overline{L_1} = \{1, 10, 11, 100, 101, \cdots\}$

Ejemplo 2. Sea L el conjunto de letras $\{A, B, C \ldots Z, a, b, c, \ldots z\}$ y D el conjunto de dígitos $\{0, 1, \ldots, 9\}$.

1. $L \cup D$ es el conjunto de letras y de números, tenemos 62 símbolos $(26+26+10) = 62$ cadenas de longitud 1. Cada una de estas cadenas es una letra o un dígito.
2. LD es el conjunto de 520 cadenas de longitud 2 $(26 + 26) * 2$, cada una siendo una letra concatenada a un número.
3. L^4 son todas las cadenas con 4 letras.
4. L^* son todas las cadenas de letras incluyendo la cadena vacía λ
5. $L(L \cup D)^*$ es el conjunto de todas las cadenas de letras y números que empiezan con una letra.
6. D^+ es el conjunto de todas las cadenas de uno o más dígitos.

3.2.3. Propiedades de las operaciones sobre lenguajes

- **Conmutatividad de la unión:** $L_1 \cup L_2 = L_2 \cup L_1$
- **Asociatividad de la unión:** $(L_1 \cup L_2) \cup L_3 = L_1 \cup (L_2 \cup L_3)$
- **Conmutatividad de la intersección:** $L_1 \cap L_2 = L_2 \cap L_1$
- **Asociatividad de la intersección:** $(L_1 \cap L_2) \cap L_3 = L_1 \cap (L_2 \cap L_3)$
- **Distributividad de la intersección sobre la unión:** $L_1 \cap (L_2 \cup L_3) = (L_1 \cap L_2) \cup (L_1 \cap L_3)$
- **Distributividad de la unión sobre la intersección:** $L_1 \cup (L_2 \cap L_3) = (L_1 \cup L_2) \cap (L_1 \cup L_3)$
- **Identidad de la unión:** $L \cup \varnothing = \varnothing \cup L = L$
- **Identidad de la concatenación:** $L \cdot \lambda = \lambda \cdot L = L$
- **El aniquilador:** $L \cdot \varnothing = \varnothing \cdot L = \varnothing$
- **Distributividad de la concatenación(1):** $A(B \cup C) = (AB) \cup (AC)$
- **Distributividad de la concatenación(2):** $(A \cup B)C = (AC) \cup (BC)$
- **Ley de Morgan(1):** $\sim (L_1 \cup L_2) = \sim L_1 \cap \sim L_2$
- **Ley de Morgan(2):** $\sim (L_1 \cap L_2) = \sim L_1 \cup \sim L_2$
- **Propiedades de la cerradura de Kleene(1):** $L^* L^* = L^*$
- **Propiedades de la cerradura de Kleene(2):** $L^{**} = L^*$
- **Propiedades de la cerradura de Kleene(3):** $L^* = \{\lambda\} \cup LL^* = \lambda \cup L^* L$
- **Propiedades de la cerradura de Kleene(4):** $\varnothing^* = \{\lambda\}$
- **Propiedades de la cerradura de Kleene(5):** $L^+ = LL^* = L^* L$

La concatenación no se distribuye sobre la intersección, por ejemplo si $A = \{a, ab\}$, $B = \{b\}$ y $C = \{\epsilon\}$, entonces $A(B \cap C) = \{a, ab\}(\{b\} \cap \{\epsilon\}) = \{a, ab\}\varnothing = \varnothing$ pero $(AB) \cap (AC) = \{ab, abb\} \cap \{a, ab\} = ab$. Pero $\varnothing \neq ab$ Contradicción.

4. Patrones y Expresiones Regulares

4.1. Patrones

Hasta ahora hemos visto que los lenguajes generan ciertas cadenas que cumplen un patrón específico.

Definición 14

Un **patrón** α es una cadena de símbolos que representan un subconjunto de elementos de Σ^*. Para representar el lenguaje que describe un patrón lo denotaremos como:

$$L(\alpha) = \{x \in \Sigma^* \mid x \text{ corresponde con } \alpha\}$$

Ahora veremos como describir estos patrones a partir de los símbolos y de las mismas operaciones que utilizamos para los lenguajes (unión, concatenación y la clausura de Kleene).

Definición 15

Tenemos **patrones atómicos**, es decir que corresponden con un símbolo;

- w para cada elemento en Σ. Es decir, este patrón corresponde con el lenguaje $L(a) = \{a\}$
- λ, corresponde con el lenguaje que solo incluye a la cadena vacía $L(\lambda) = \{\lambda\}$
- $\varnothing$, corresponde al lenguaje vacío $L(\varnothing) = \varnothing$

Y tenemos **patrones compuestos**, que se forman inductivamente a partir de patrones atómicos. Si α y β son patrones, entonces podemos formar nuevos patrones de la siguiente manera:

- **Unión:** $\alpha \cup \beta$ es un patrón que corresponde con el lenguaje $L(\alpha) \cup L(\beta)$.
- **Concatenación:** $\alpha \cdot \beta$ es un patrón que corresponde con el lenguaje $L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$.
- **Clausura de Kleene:** α^* es un patrón que corresponde con el lenguaje

$$L(\alpha^*) = L^0(\alpha) \cup L^1(\alpha) \cup L^2(\alpha) \cup \dots$$

.

- **Intersección:** $\alpha \cap \beta$ es un patrón que corresponde con el lenguaje $L(\alpha \cap \beta) = L(\alpha) \cap L(\beta)$.
- **Negación:** $\neg\alpha$ es un patrón que corresponde con el lenguaje $L(\neg\alpha) = \Sigma^* - L(\alpha)$.

En $UNIX$ podemos ocupar el comando `grep` para encontrar cadenas que representen cierto patrón. Y aquí tenemos símbolos que tienen una función específica, como;

- `.` - sirve para representar cualquier carácter
- `^` - representa el inicio de una línea
- `+` - representa uno o más que la expresión se encuentra una o más veces (similar a la clausura positiva)
- `*` - representa que la cadena se encuentra cero o más caracteres (clausura de Kleene)
- `?` - representa que la cadena se encuentra cero o una vez
- `d` - representa cualquier dígito (0-9)
- `D` - representa cualquier carácter que no sea un dígito
- `s` - representa cualquier espacio en blanco (espacio, tabulación, salto de línea)

Otros símbolos útiles incluyen:

- ‘`[a-z]`‘ - representa cualquier carácter dentro de los corchetes, este caso el conjunto de todas las letras minusculas.
- ‘[^0-1]‘ - representa cualquier carácter que no esté dentro de los corchetes. En este caso el conjunto de todos los caracteres que no tienen un número.

Intenta ver que se imprimia en la terminal con estos comandos.

```
echo abcd | grep ^ab
echo abcd | grep ab$
echo "123_456_123_456" | grep -E '([0-9]+)_([0-9]+)_\1_\2'
echo "777 476 1820" | grep -E '^\([0-9]{3}\) [0-9]{3}-[0-9]{4}$'
```

4.2. Expresiones Regulares

Las **expresiones regulares** son otra manera de definir lenguajes, muy parecidos a los patrones **patrones**. Aunque todavía no hemos visto los Autómatas, más adelante veremos que las expresiones regulares son una manera concisa de representar el lenguaje que genera un automata (de hecho, probaremos que son equivalentes y veremos un algoritmo para pasar de expresiones regulares a autómatas y viceversa). Y son una representación más fácil programar para poder construir compiladores o lenguajes de programación.

Definición 16

Decimos que R es un **expresión regular** si R es:

1. w para algun símbolo en Σ
2. λ, la cadena vacía
3. $\varnothing$
4. $R_1 + R_2$, donde $R1$ y R_2 son expresiones regulares
5. R_1R_2, donde $R1$ y R_2 son expresiones regulares
6. R_1^*, donde R_1 es una expresión regular

Ejemplos:

1. Si $\Sigma = \{\texttt{a,b,c,...}\}$, entonces `hola` y `holi` son expresiones regulares.

$$L(\texttt{hola}) = \texttt{hola}$$
$$L(\texttt{holi}) = \texttt{holi}$$
$$L(\texttt{hol(a+i)}) = \{\texttt{hola,holi}\}$$
$$L(\texttt{ho}^{+}\texttt{l(a+i)}) = \{\texttt{hola, holi, hoola,hooli, ...}\}$$

   Para los siguientes ejemplos consideremos el alfabeto $\Sigma = \{0, 1\}$

2. $(0 + 1)^* = \{0, 00, 000, 000, \dots\} \cup \{1, 11, 111, 1111, \dots\}$, es decir todas cadenas de $0's$ o de $1's$
3. $\Sigma^*1\Sigma = \{1,01,010,110,111, \dots\}$, son todas las cadenas que tienen al menos un uno.
4. $\Sigma^*001\Sigma^*$, son todas las cadenas que tienen 001 como subcadena.
5. $(\Sigma\Sigma)^*$, todas las cadenas de tamaño par.
6. $01 = \{01\}$
7. $(0 + \lambda)1 = 01 + 1$
8. $1\varnothing = \varnothing$
9. $\varnothing^* = \{\lambda\}$

   Podemos intentar encajar un patron de teléfono o un número decimal o entero en una expresión regular. Por ejemplo si, $D = \{0, 1, \dots 9\}$

$$(+ \cup - \cup \lambda)((D^*) \cup (D^+, D^*) \cup (D^*.D^+))$$

   - $D*$ son todos los enteros, (eg. 43)
   - $(D^+.D^*)$ son todos los números decimales, con un dígito enfrente (e.g 3,1416)
   - $(D^*.D^+)$ son todos los números decimales, puede ser ,0111

   Más adelante cuando veamos como pasar expresiones regulares a autómatas veremos que es más fácil tratar de minimizar primero la expresión. Para ello existen algunas reglas algebraicas con las que podemos hacer operaciones.