

# Autómatas y Lenguajes Formales

Maestría en Ciencia e Ingeniería de la Computación UNAM  
Sesión 1: Introducción

Dr. Favio Ezequiel Miranda Perea  
`favio@ciencias.unam.mx`

Facultad de Ciencias UNAM<sup>1</sup>

27 de abril de 2019



---

<sup>1</sup>Con el apoyo del proyecto PAPIME PE102117

# Introducción

## Fundamentos de la Computación

- ¿ Qué son los fundamentos de la computación?



# Introducción

## Fundamentos de la Computación

- ¿ Qué son los fundamentos de la computación?
- Ciencias de la computación: conglomerado de disciplinas científicas y de ingeniería relacionadas con el estudio y aplicación del cómputo.



# Introducción

## Fundamentos de la Computación

- ¿ Qué son los fundamentos de la computación?
- Ciencias de la computación: conglomerado de disciplinas científicas y de ingeniería relacionadas con el estudio y aplicación del cómputo.
- Desde las mas puras y básicas dedicadas a los fundamentos de la computación.



# Introducción

## Fundamentos de la Computación

- ¿ Qué son los fundamentos de la computación?
- Ciencias de la computación: conglomerado de disciplinas científicas y de ingeniería relacionadas con el estudio y aplicación del cómputo.
- Desde las mas puras y básicas dedicadas a los fundamentos de la computación.
- Hasta las ingenierías dedicadas a aplicaciones específicas.



# Fundamentos de la Computación

Los fundamentos de la computación se dividen esencialmente en dos:

- Teoría de la Programación:



# Fundamentos de la Computación

Los fundamentos de la computación se dividen esencialmente en dos:

- Teoría de la Programación:

Dedicada a estudiar los lenguajes de programación para implementar cómputos.



# Fundamentos de la Computación

Los fundamentos de la computación se dividen esencialmente en dos:

- Teoría de la Programación:

Dedicada a estudiar los lenguajes de programación para implementar cálculos.

- Teoría de la Computación:



# Fundamentos de la Computación

Los fundamentos de la computación se dividen esencialmente en dos:

- Teoría de la Programación:

Dedicada a estudiar los lenguajes de programación para implementar cómputos.

- Teoría de la Computación:

Dedicada a entender la naturaleza del cómputo, sus posibilidades y limitaciones.



# Teoría de la Programación

Se divide en diversas disciplinas:

- Teoría de Lenguajes de Programación



# Teoría de la Programación

Se divide en diversas disciplinas:

- Teoría de Lenguajes de Programación
- Semántica de Lenguajes



# Teoría de la Programación

Se divide en diversas disciplinas:

- Teoría de Lenguajes de Programación
- Semántica de Lenguajes
- Teoría de Tipos



# Teoría de la Programación

Se divide en diversas disciplinas:

- Teoría de Lenguajes de Programación
- Semántica de Lenguajes
- Teoría de Tipos
- Estilos de Programación: lógica, funcional, imperativa,...



# Teoría de la Programación

Se divide en diversas disciplinas:

- Teoría de Lenguajes de Programación
- Semántica de Lenguajes
- Teoría de Tipos
- Estilos de Programación: lógica, funcional, imperativa,...
- Especificación y Verificación



# Teoría de la Programación

Se divide en diversas disciplinas:

- Teoría de Lenguajes de Programación
- Semántica de Lenguajes
- Teoría de Tipos
- Estilos de Programación: lógica, funcional, imperativa,...
- Especificación y Verificación
- Lógica Computacional



# Teoría de la Computación

Mediante la teoría del cómputo podemos tratar preguntas como:

- ¿Qué es un dispositivo de cómputo?



# Teoría de la Computación

Mediante la teoría del cómputo podemos tratar preguntas como:

- ¿Qué es un dispositivo de cómputo?
- ¿Qué se puede computar?



# Teoría de la Computación

Mediante la teoría del cómputo podemos tratar preguntas como:

- ¿Qué es un dispositivo de cómputo?
- ¿Qué se puede computar?
- ¿Qué **no** se puede computar?



# Teoría de la Computación

Mediante la teoría del cómputo podemos tratar preguntas como:

- ¿Qué es un dispositivo de cómputo?
- ¿Qué se puede computar?
- ¿Qué **no** se puede computar?
- ¿Cual es el costo de un cómputo?



# Teoría de la Computación

Mediante la teoría del cómputo podemos tratar preguntas como:

- ¿Qué es un dispositivo de cómputo?
- ¿Qué se puede computar?
- ¿Qué **no** se puede computar?
- ¿Cual es el costo de un cómputo?
- ¿Qué se puede computar **eficientemente**?



# Teoría de la Computación

Mediante la teoría del cómputo podemos tratar preguntas como:

- ¿Qué es un dispositivo de cómputo?
- ¿Qué se puede computar?
- ¿Qué **no** se puede computar?
- ¿Cual es el costo de un cómputo?
- ¿Qué se puede computar **eficientemente**?
- ¿Como clasificar un problema de acuerdo a su dificultad?



# Fundamentos de la Computación

Nosotros nos dedicaremos exclusivamente a dos aspectos de la teoría de la computación:

- Teoría de Autómatas y Lenguajes Formales



# Fundamentos de la Computación

Nosotros nos dedicaremos exclusivamente a dos aspectos de la teoría de la computación:

- Teoría de Autómatas y Lenguajes Formales
- Introducción al análisis de algoritmos.



# Importancia de la Teoría de la Computación

## Introducción

- Entender cuales son las capacidades y limitaciones fundamentales de una computadora.



# Importancia de la Teoría de la Computación

## Introducción

- Entender cuales son las capacidades y limitaciones fundamentales de una computadora.
- Captura de las nociones de *cómputo* y *computabilidad efectiva* mediante abstracción matemática.



# Importancia de la Teoría de la Computación

## Introducción

- Entender cuales son las capacidades y limitaciones fundamentales de una computadora.
- Captura de las nociones de *cómputo* y *computabilidad efectiva* mediante abstracción matemática.
- Modelación mediante autómatas:



# Importancia de la Teoría de la Computación

## Introducción

- Entender cuales son las capacidades y limitaciones fundamentales de una computadora.
- Captura de las nociones de *cómputo* y *computabilidad efectiva* mediante abstracción matemática.
- Modelación mediante autómatas:
  - ▶ Diseño de circuitos digitales.



# Importancia de la Teoría de la Computación

## Introducción

- Entender cuales son las capacidades y limitaciones fundamentales de una computadora.
- Captura de las nociones de *cómputo* y *computabilidad efectiva* mediante abstracción matemática.
- Modelación mediante autómatas:
  - ▶ Diseño de circuitos digitales.
  - ▶ Analizadores léxicos para compiladores.



# Importancia de la Teoría de la Computación

## Introducción

- Entender cuales son las capacidades y limitaciones fundamentales de una computadora.
- Captura de las nociones de *cómputo* y *computabilidad efectiva* mediante abstracción matemática.
- Modelación mediante autómatas:
  - ▶ Diseño de circuitos digitales.
  - ▶ Analizadores léxicos para compiladores.
  - ▶ Búsqueda de palabras clave en internet.



# Importancia de la Teoría de la Computación

## Introducción

- Entender cuales son las capacidades y limitaciones fundamentales de una computadora.
- Captura de las nociones de *cómputo* y *computabilidad efectiva* mediante abstracción matemática.
- Modelación mediante autómatas:
  - ▶ Diseño de circuitos digitales.
  - ▶ Analizadores léxicos para compiladores.
  - ▶ Búsqueda de palabras clave en internet.
  - ▶ Verificación de sistemas de estados finitos (por ejemplo protocolos de comunicación)



# Abstracción del Concepto de Computadora

## Introducción

- Una computadora es una máquina que transforma ciertos datos de entrada dados en resultados o datos de salida.



# Abstracción del Concepto de Computadora

## Introducción

- Una computadora es una máquina que transforma ciertos datos de entrada dados en resultados o datos de salida.
- Podemos pensar que una computadora es una *función* que transforma sus argumentos de entrada en resultados.



# Abstracción del Concepto de Computadora

## Introducción

- Una computadora es una máquina que transforma ciertos datos de entrada dados en resultados o datos de salida.
- Podemos pensar que una computadora es una *función* que transforma sus argumentos de entrada en resultados.
- ¿ Cual sería el dominio de tal función? es decir, que tipos de datos se esperan como entrada: números, palabras, textos, imagenes, etc.



# Abstracción del Concepto de Computadora

## Introducción

- Una computadora es una máquina que transforma ciertos datos de entrada dados en resultados o datos de salida.
- Podemos pensar que una computadora es una *función* que transforma sus argumentos de entrada en resultados.
- ¿ Cual sería el dominio de tal función? es decir, que tipos de datos se esperan como entrada: números, palabras, textos, imagenes, etc.



# Abstracción del Concepto de Computadora

## Introducción

- Una computadora es una máquina que transforma ciertos datos de entrada dados en resultados o datos de salida.
- Podemos pensar que una computadora es una *función* que transforma sus argumentos de entrada en resultados.
- ¿ Cual sería el dominio de tal función? es decir, que tipos de datos se esperan como entrada: números, palabras, textos, imagenes, etc.

¿ Cómo podemos representar los datos de entrada de manera uniforme ?



# Abstracción del Concepto de Computadora

## Idea Básica

Cualesquiera datos de entrada para una computadora pueden ser codificados mediante una cadena o sucesión de símbolos.



# Abstracción del Concepto de Computadora

## Idea Básica

Cualesquiera datos de entrada para una computadora pueden ser codificados mediante una cadena o sucesión de símbolos.

- ¿Qué cadenas son aceptables como entrada ?



# Abstracción del Concepto de Computadora

## Idea Básica

Cualesquiera datos de entrada para una computadora pueden ser codificados mediante una cadena o sucesión de símbolos.

- ¿Qué cadenas son aceptables como entrada ?
- ¿Qué cadenas se obtienen como salida ?



# Abstracción del Concepto de Computadora

## Idea Básica

Cualesquiera datos de entrada para una computadora pueden ser codificados mediante una cadena o sucesión de símbolos.

- ¿Qué cadenas son aceptables como entrada ?
- ¿Qué cadenas se obtienen como salida ?
- ¿Será posible caracterizar de manera finita a estos conjuntos de cadenas, los cuales pueden ser infinitos?



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*

**Ejemplos:**

#



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*

**Ejemplos:**

#, %



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*

**Ejemplos:**

#, %, \$



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*

**Ejemplos:**

#, %, \$, ←



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*

**Ejemplos:**

#, %, \$, ← , ^



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*

**Ejemplos:**

#, %, \$,  $\leftarrow$ ,  $\wedge$ , *a*



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*

**Ejemplos:**

#, %, \$,  $\leftarrow$ ,  $\wedge$ , *a*, 7



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*

## Ejemplos:

#, %, \$,  $\leftarrow$ ,  $\wedge$ , *a*, 7

Por comodidad utilizaremos como símbolos:



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*

## Ejemplos:

#, %, \$,  $\leftarrow$ ,  $\wedge$ , *a*, 7

Por comodidad utilizaremos como símbolos:

- *a,b,c,d,e,...*



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*

## Ejemplos:

#, %, \$,  $\leftarrow$ ,  $\wedge$ , *a*, 7

Por comodidad utilizaremos como símbolos:

- *a,b,c,d,e,...*



# ¿ Qué es un símbolo ?

Los **símbolos** son los objetos más simples con los que trataremos:

Un **símbolo** o caracter es *una entidad indivisible*

## Ejemplos:

$\#$ ,  $\%$ ,  $\$$ ,  $\leftarrow$ ,  $\wedge$ ,  $a$ ,  $7$

Por comodidad utilizaremos como símbolos:

- $a, b, c, d, e, \dots$
- $0, 1, 2, 3, \dots, 9$



# Alfabetos

Un **alfabeto** es un conjunto *finito* de símbolos.



# Alfabetos

Un **alfabeto** es un conjunto *finito* de símbolos.

## Ejemplos:

- El alfabeto del español:  $a, b, c, d, e, f, \dots, z$ .



# Alfabetos

Un **alfabeto** es un conjunto *finito* de símbolos.

## Ejemplos:

- El alfabeto del español:  $a, b, c, d, e, f, \dots, z$ .
- El alfabeto binario:  $0, 1$ .



# Alfabetos

Un **alfabeto** es un conjunto *finito* de símbolos.

## Ejemplos:

- El alfabeto del español:  $a, b, c, d, e, f, \dots, z$ .
- El alfabeto binario:  $0, 1$ .
- El alfabeto ASCII:  $a, \dots, z, A, \dots, Z, \$, \%, \#, \dots$



# Alfabetos

Un **alfabeto** es un conjunto *finito* de símbolos.

## Ejemplos:

- El alfabeto del español:  $a, b, c, d, e, f, \dots, z$ .
- El alfabeto binario:  $0, 1$ .
- El alfabeto ASCII:  $a, \dots, z, A, \dots, Z, \$, \%, \#, \dots$

Usaremos la letra griega  $\Sigma$  o letras mayúsculas del final del alfabeto ( $X, Y, Z$ ) para denotar alfabetos.



# Cadenas

Una **cadena** o *palabra* es una sucesión finita de símbolos tomados de un alfabeto dado.

**Ejemplos:**



# Cadenas

Una **cadena** o *palabra* es una sucesión finita de símbolos tomados de un alfabeto dado.

## Ejemplos:

- En el alfabeto del español:  
*abc, def, feo, bonito, dsp, guajolote, uizcm.*



# Cadenas

Una **cadena** o *palabra* es una sucesión finita de símbolos tomados de un alfabeto dado.

## Ejemplos:

- En el alfabeto del español:  
*abc, def, feo, bonito, dsp, guajolote, uizcm.*
- En el alfabeto binario: 0, 101010, 00, 1100, 001, 11010.



# Cadenas

Una **cadena** o *palabra* es una sucesión finita de símbolos tomados de un alfabeto dado.

## Ejemplos:

- En el alfabeto del español:  
*abc, def, feo, bonito, dsp, guajolote, uizcm.*
- En el alfabeto binario: 0, 101010, 00, 1100, 001, 11010.

Obsérvese que los símbolos son a su vez cadenas que constan de un solo caracter. Más aún la **cadena vacía** (i.e. la sucesión vacía de símbolos) se denotará con  $\epsilon$ .



# Cadenas

Una **cadena** o *palabra* es una sucesión finita de símbolos tomados de un alfabeto dado.

## Ejemplos:

- En el alfabeto del español:  
*abc, def, feo, bonito, dsp, guajolote, uizcm.*
- En el alfabeto binario: 0, 101010, 00, 1100, 001, 11010.

Obsérvese que los símbolos son a su vez cadenas que constan de un solo caracter. Más aún la **cadena vacía** (i.e. la sucesión vacía de símbolos) se denotará con  $\epsilon$ .

Al conjunto infinito de todas las cadenas sobre  $\Sigma$  se le denota con  $\Sigma^*$



# Longitud y Concatenación

- La **longitud** de una cadena  $w$  es el número de símbolos en  $w$  y se denota con  $|w|$ .



# Longitud y Concatenación

- La **longitud** de una cadena  $w$  es el número de símbolos en  $w$  y se denota con  $|w|$ .
- La operación básica entre cadenas es la **concatenación** que consiste en pegar cadenas en orden de izquierda a derecha: Si  $v, w$  son cadenas entonces  $vw$  será la cadena obtenida al pegar  $v$  con  $w$ :



# Longitud y Concatenación

- La **longitud** de una cadena  $w$  es el número de símbolos en  $w$  y se denota con  $|w|$ .
- La operación básica entre cadenas es la **concatenación** que consiste en pegar cadenas en orden de izquierda a derecha: Si  $v, w$  son cadenas entonces  $vw$  será la cadena obtenida al pegar  $v$  con  $w$ :
  - ▶ La concatenación de *cala* y *baza* es la cadena *calabaza*.



# Longitud y Concatenación

- La **longitud** de una cadena  $w$  es el número de símbolos en  $w$  y se denota con  $|w|$ .
- La operación básica entre cadenas es la **concatenación** que consiste en pegar cadenas en orden de izquierda a derecha: Si  $v, w$  son cadenas entonces  $vw$  será la cadena obtenida al pegar  $v$  con  $w$ :
  - ▶ La concatenación de *cala* y *baza* es la cadena *calabaza*.
  - ▶ si  $v = \textit{broco}$  y  $w = \textit{li}$  entonces  $vw = \textit{brocoli}$ .



# Longitud y Concatenación

- La **longitud** de una cadena  $w$  es el número de símbolos en  $w$  y se denota con  $|w|$ .
- La operación básica entre cadenas es la **concatenación** que consiste en pegar cadenas en orden de izquierda a derecha: Si  $v, w$  son cadenas entonces  $vw$  será la cadena obtenida al pegar  $v$  con  $w$ :
  - ▶ La concatenación de *cala* y *baza* es la cadena *calabaza*.
  - ▶ si  $v = \textit{broco}$  y  $w = \textit{li}$  entonces  $vw = \textit{brocoli}$ .
  - ▶ si  $x = \textit{champu}$  y  $y = \textit{rrado}$  entonces  $yx = \textit{rradochampu}$ .



# Propiedades de la Concatenación

- Asociatividad:  $(uv)w = u(vw)$ .



# Propiedades de la Concatenación

- Asociatividad:  $(uv)w = u(vw)$ .
- Identidad:  $v\varepsilon = \varepsilon v = v$ .



# Propiedades de la Concatenación

- Asociatividad:  $(uv)w = u(vw)$ .
- Identidad:  $v\varepsilon = \varepsilon v = v$ .
- Longitud:  $|vw| = |v| + |w|$ .



# Propiedades de la Concatenación

- Asociatividad:  $(uv)w = u(vw)$ .
- Identidad:  $v\varepsilon = \varepsilon v = v$ .
- Longitud:  $|vw| = |v| + |w|$ .

¿ Es conmutativa la concatenación ?



# Propiedades de la Concatenación

- Asociatividad:  $(uv)w = u(vw)$ .
- Identidad:  $v\varepsilon = \varepsilon v = v$ .
- Longitud:  $|vw| = |v| + |w|$ .

¿ Es conmutativa la concatenación ?



# Propiedades de la Concatenación

- Asociatividad:  $(uv)w = u(vw)$ .
- Identidad:  $v\varepsilon = \varepsilon v = v$ .
- Longitud:  $|vw| = |v| + |w|$ .

¿ Es conmutativa la concatenación ?

¿  $vw = wv$  ?



# Propiedades de la Concatenación

- Asociatividad:  $(uv)w = u(vw)$ .
- Identidad:  $v\varepsilon = \varepsilon v = v$ .
- Longitud:  $|vw| = |v| + |w|$ .

¿ Es conmutativa la concatenación ?

¿  $vw = wv$  ?



# Propiedades de la Concatenación

- Asociatividad:  $(uv)w = u(vw)$ .
- Identidad:  $v\varepsilon = \varepsilon v = v$ .
- Longitud:  $|vw| = |v| + |w|$ .

¿ Es conmutativa la concatenación ?

¿  $vw = wv$  ?

NO *jarrito*  $\neq$  *rritoja*



# Reversa

La reversa de una cadena  $u$ , denotada  $u^R$ , se define como sigue:

Si  $u = a_1 a_2 \dots a_n$  entonces  $u^R = a_n a_{n-1} \dots a_2 a_1$ .



# Reversa

La reversa de una cadena  $u$ , denotada  $u^R$ , se define como sigue:

Si  $u = a_1 a_2 \dots a_n$  entonces  $u^R = a_n a_{n-1} \dots a_2 a_1$ .



# Reversa

La reversa de una cadena  $u$ , denotada  $u^R$ , se define como sigue:

Si  $u = a_1 a_2 \dots a_n$  entonces  $u^R = a_n a_{n-1} \dots a_2 a_1$ .

Se cumplen las siguientes propiedades:

- $(u^R)^R = u$



# Reversa

La reversa de una cadena  $u$ , denotada  $u^R$ , se define como sigue:

Si  $u = a_1 a_2 \dots a_n$  entonces  $u^R = a_n a_{n-1} \dots a_2 a_1$ .

Se cumplen las siguientes propiedades:

- $(u^R)^R = u$
- $(uv)^R = v^R u^R$



# Subcadenas, Prefijos y Sufijos

- Decimos que  $v$  es una subcadena de  $u$  si existen cadenas  $x, y \in \Sigma^*$  tales que  $u = xvy$ .



# Subcadenas, Prefijos y Sufijos

- Decimos que  $v$  es una subcadena de  $u$  si existen cadenas  $x, y \in \Sigma^*$  tales que  $u = xvy$ .
- Un prefijo de  $u$  es una cadena  $v$  tal que  $u = vw$  para alguna cadena  $w \in \Sigma^*$ . Se dice que  $v$  es un prefijo propio si  $v \neq u$ .



# Subcadenas, Prefijos y Sufijos

- Decimos que  $v$  es una subcadena de  $u$  si existen cadenas  $x, y \in \Sigma^*$  tales que  $u = xvy$ .
- Un prefijo de  $u$  es una cadena  $v$  tal que  $u = vw$  para alguna cadena  $w \in \Sigma^*$ . Se dice que  $v$  es un prefijo propio si  $v \neq u$ .
- Similarmente, un sufijo de  $u$  es una cadena  $v$  tal que  $u = wv$  para alguna cadena  $w \in \Sigma^*$ . Se dice que  $v$  es un sufijo propio si  $v \neq u$ .



# Subcadenas, Prefijos y Sufijos

- Decimos que  $v$  es una subcadena de  $u$  si existen cadenas  $x, y \in \Sigma^*$  tales que  $u = xvy$ .
- Un prefijo de  $u$  es una cadena  $v$  tal que  $u = vw$  para alguna cadena  $w \in \Sigma^*$ . Se dice que  $v$  es un prefijo propio si  $v \neq u$ .
- Similarmente, un sufijo de  $u$  es una cadena  $v$  tal que  $u = wv$  para alguna cadena  $w \in \Sigma^*$ . Se dice que  $v$  es un sufijo propio si  $v \neq u$ .
- Obsérvese que tanto  $\varepsilon$  como  $u$  son siempre sufijos y prefijos de  $u$ .



# Definición de Lenguaje



# Definición de Lenguaje

Un **lenguaje**  $\mathcal{L}$  sobre un alfabeto  $\Sigma$  es simplemente un conjunto de cadenas de  $\Sigma$ . Es decir  $\mathcal{L} \subseteq \Sigma^*$ .



# Definición de Lenguaje

Un **lenguaje**  $\mathcal{L}$  sobre un alfabeto  $\Sigma$  es simplemente un conjunto de cadenas de  $\Sigma$ . Es decir  $\mathcal{L} \subseteq \Sigma^*$ .

Si  $\Sigma = \{m, u\}$  entonces algunos lenguajes sobre  $\Sigma$  son:

- $\mathcal{L}_1 = \{m, u\} = \Sigma$



# Definición de Lenguaje

Un **lenguaje**  $\mathcal{L}$  sobre un alfabeto  $\Sigma$  es simplemente un conjunto de cadenas de  $\Sigma$ . Es decir  $\mathcal{L} \subseteq \Sigma^*$ .

Si  $\Sigma = \{m, u\}$  entonces algunos lenguajes sobre  $\Sigma$  son:

- $\mathcal{L}_1 = \{m, u\} = \Sigma$
- $\mathcal{L}_2 = \{u, uu, uuu, uuuu, uuuuu, uuuuuu, \dots\}.$



# Definición de Lenguaje

Un **lenguaje**  $\mathcal{L}$  sobre un alfabeto  $\Sigma$  es simplemente un conjunto de cadenas de  $\Sigma$ . Es decir  $\mathcal{L} \subseteq \Sigma^*$ .

Si  $\Sigma = \{m, u\}$  entonces algunos lenguajes sobre  $\Sigma$  son:

- $\mathcal{L}_1 = \{m, u\} = \Sigma$
- $\mathcal{L}_2 = \{u, uu, uuu, uuuu, uuuuu, uuuuuu, \dots\}$ .
- $\mathcal{L}_3 = \{mu, um, mum, umu, mmu, ummm, mumumum\}$ .



# Definición de Lenguaje

Un **lenguaje**  $\mathcal{L}$  sobre un alfabeto  $\Sigma$  es simplemente un conjunto de cadenas de  $\Sigma$ . Es decir  $\mathcal{L} \subseteq \Sigma^*$ .

Si  $\Sigma = \{m, u\}$  entonces algunos lenguajes sobre  $\Sigma$  son:

- $\mathcal{L}_1 = \{m, u\} = \Sigma$
- $\mathcal{L}_2 = \{u, uu, uuu, uuuu, uuuuu, uuuuuu, \dots\}$ .
- $\mathcal{L}_3 = \{mu, um, mum, umu, mmu, ummm, mumumum\}$ .
- $\mathcal{L}_4 = \{mu, muu, muuu, muuuu, \dots\}$ .



# Definición de Lenguaje

Un **lenguaje**  $\mathcal{L}$  sobre un alfabeto  $\Sigma$  es simplemente un conjunto de cadenas de  $\Sigma$ . Es decir  $\mathcal{L} \subseteq \Sigma^*$ .

Si  $\Sigma = \{m, u\}$  entonces algunos lenguajes sobre  $\Sigma$  son:

- $\mathcal{L}_1 = \{m, u\} = \Sigma$
- $\mathcal{L}_2 = \{u, uu, uuu, uuuu, uuuuu, uuuuuu, \dots\}$ .
- $\mathcal{L}_3 = \{mu, um, mum, umu, mmu, ummm, mumumum\}$ .
- $\mathcal{L}_4 = \{mu, muu, muuu, muuuu, \dots\}$ .
- $\mathcal{L}_5 = \{m, mmm, mmmmm, \dots, m\dots, uuum, m\dots\}$ .



# Operaciones con Lenguajes

Dado que los lenguajes son conjuntos todas las operaciones conjuntistas son aplicables a lenguajes. Si  $L, M \subseteq \Sigma^*$  entonces

$$L \cup M,$$



# Operaciones con Lenguajes

Dado que los lenguajes son conjuntos todas las operaciones conjuntistas son aplicables a lenguajes. Si  $L, M \subseteq \Sigma^*$  entonces

$$L \cup M, L \cap M,$$



# Operaciones con Lenguajes

Dado que los lenguajes son conjuntos todas las operaciones conjuntistas son aplicables a lenguajes. Si  $L, M \subseteq \Sigma^*$  entonces

$$L \cup M, L \cap M, L - M,$$



# Operaciones con Lenguajes

Dado que los lenguajes son conjuntos todas las operaciones conjuntistas son aplicables a lenguajes. Si  $L, M \subseteq \Sigma^*$  entonces

$$L \cup M, L \cap M, L - M, \bar{L} = \Sigma^* - L$$

tambien son lenguajes.



# Concatenación de Lenguajes

Al igual que en el caso de cadenas, podemos definir la concatenación entre lenguajes como sigue:

$$LM = \{uv \mid u \in L \text{ y } v \in M\}$$

Se cumplen las siguientes propiedades:

- $L\emptyset = \emptyset L = \emptyset$



# Concatenación de Lenguajes

Al igual que en el caso de cadenas, podemos definir la concatenación entre lenguajes como sigue:

$$LM = \{uv \mid u \in L \text{ y } v \in M\}$$

Se cumplen las siguientes propiedades:

- $L\emptyset = \emptyset L = \emptyset$
- $L\{\varepsilon\} = \{\varepsilon\}L = L$



# Concatenación de Lenguajes

Al igual que en el caso de cadenas, podemos definir la concatenación entre lenguajes como sigue:

$$LM = \{uv \mid u \in L \text{ y } v \in M\}$$

Se cumplen las siguientes propiedades:

- $L\emptyset = \emptyset L = \emptyset$
- $L\{\varepsilon\} = \{\varepsilon\}L = L$
- $L(MN) = (LM)N$



# Concatenación de Lenguajes

Al igual que en el caso de cadenas, podemos definir la concatenación entre lenguajes como sigue:

$$LM = \{uv \mid u \in L \text{ y } v \in M\}$$

Se cumplen las siguientes propiedades:

- $L\emptyset = \emptyset L = \emptyset$
- $L\{\varepsilon\} = \{\varepsilon\}L = L$
- $L(MN) = (LM)N$
- $L(M \cup N) = LM \cup LN \quad (M \cup N)L = ML \cup NL$



# Reversa de un Lenguaje

La reversa de un lenguaje se define como

$$L^R = \{u^R \mid u \in L\}$$

Se cumplen las siguientes propiedades:

- $(L^R)^R = L$



# Reversa de un Lenguaje

La reversa de un lenguaje se define como

$$L^R = \{u^R \mid u \in L\}$$

Se cumplen las siguientes propiedades:

- $(L^R)^R = L$
- $(LM)^R = M^R L^R$



# Reversa de un Lenguaje

La reversa de un lenguaje se define como

$$L^R = \{u^R \mid u \in L\}$$

Se cumplen las siguientes propiedades:

- $(L^R)^R = L$
- $(LM)^R = M^R L^R$
- $(L \cup M)^R = L^R \cup M^R$



# Reversa de un Lenguaje

La reversa de un lenguaje se define como

$$L^R = \{u^R \mid u \in L\}$$

Se cumplen las siguientes propiedades:

- $(L^R)^R = L$
- $(LM)^R = M^R L^R$
- $(L \cup M)^R = L^R \cup M^R$
- $(L \cap M)^R = L^R \cap M^R$



# Cerradura de Kleene

La cerradura o estrella de Kleene de un lenguaje se define como

$$L^* = \{u_1 \dots u_n \mid u_i \in L, n \geq 0\} = \bigcup_{i=0}^{\infty} L^i$$

donde  $L^i = LL \dots L$ ,  $i$  veces.



# Cerradura Transitiva

La cerradura transitiva de un lenguaje se define como

$$L^+ = \{u_1 \dots u_n \mid u_i \in L, n > 0\} = \bigcup_{i=1}^{\infty} L^i$$

donde  $L^i = LL \dots L$ ,  $i$  veces.



# Propiedades de las cerraduras

Se cumplen las siguientes propiedades

- $L^* = L^+ \cup \{\varepsilon\}$



# Propiedades de las cerraduras

Se cumplen las siguientes propiedades

- $L^* = L^+ \cup \{\varepsilon\}$
- $L^* = L^+$  si y sólo si  $\varepsilon \in L$ .



# Propiedades de las cerraduras

Se cumplen las siguientes propiedades

- $L^* = L^+ \cup \{\varepsilon\}$
- $L^* = L^+$  si y sólo si  $\varepsilon \in L$ .
- $(L^*)^* = L^*$



# Propiedades de las cerraduras

Se cumplen las siguientes propiedades

- $L^* = L^+ \cup \{\varepsilon\}$
- $L^* = L^+$  si y sólo si  $\varepsilon \in L$ .
- $(L^*)^* = L^*$
- $L^*L^* = L^*$



# Propiedades de las cerraduras

Se cumplen las siguientes propiedades

- $L^* = L^+ \cup \{\varepsilon\}$
- $L^* = L^+$  si y sólo si  $\varepsilon \in L$ .
- $(L^*)^* = L^*$
- $L^*L^* = L^*$
- $(L^+)^* = (L^*)^+ = L^*$



# Propiedades de las cerraduras

Se cumplen las siguientes propiedades

- $L^* = L^+ \cup \{\varepsilon\}$
- $L^* = L^+$  si y sólo si  $\varepsilon \in L$ .
- $(L^*)^* = L^*$
- $L^*L^* = L^*$
- $(L^+)^* = (L^*)^+ = L^*$
- $(L^+)^+ = L^+$



# Propiedades de las cerraduras

Se cumplen las siguientes propiedades

- $L^* = L^+ \cup \{\varepsilon\}$
- $L^* = L^+$  si y sólo si  $\varepsilon \in L$ .
- $(L^*)^* = L^*$
- $L^*L^* = L^*$
- $(L^+)^* = (L^*)^+ = L^*$
- $(L^+)^+ = L^+$
- $L^+L^+ \subseteq L^+$



# Lenguajes y Computadoras

Con el concepto de lenguaje podemos reformular las preguntas acerca de los datos de entrada y salida en una computadora:



# Lenguajes y Computadoras

Con el concepto de lenguaje podemos reformular las preguntas acerca de los datos de entrada y salida en una computadora:

- ¿ Cual es el lenguaje de entrada de una computadora dada?



# Lenguajes y Computadoras

Con el concepto de lenguaje podemos reformular las preguntas acerca de los datos de entrada y salida en una computadora:

- ¿ Cual es el lenguaje de entrada de una computadora dada?
- ¿ Cual es el lenguaje de salida ?



# Lenguajes y Computadoras

Con el concepto de lenguaje podemos reformular las preguntas acerca de los datos de entrada y salida en una computadora:

- ¿ Cual es el lenguaje de entrada de una computadora dada?
- ¿ Cual es el lenguaje de salida ?
- ¿ Serán estos lenguajes descriptibles finitamente?



# Relevancia en Ciencias de la Computación

## Inducción

- Mecanismo de definición de tipos de datos



# Relevancia en Ciencias de la Computación

## Inducción

- Mecanismo de definición de tipos de datos
- Mecanismo de definición de funciones (iteración/recursión)



# Relevancia en Ciencias de la Computación

## Inducción

- Mecanismo de definición de tipos de datos
- Mecanismo de definición de funciones (iteración/recursión)
- Herramienta en correctud de programas.



# Relevancia en Ciencias de la Computación

## Inducción

- Mecanismo de definición de tipos de datos
- Mecanismo de definición de funciones (iteración/recursión)
- Herramienta en correctud de programas.
- etc etc etc



# Números Naturales

## Definiciones Recursivas

$\mathbb{N}$  es el conjunto más pequeño tal que:



# Números Naturales

## Definiciones Recursivas

$\mathbb{N}$  es el conjunto más pequeño tal que:

- $0 \in \mathbb{N}$  (condición inicial)



# Números Naturales

## Definiciones Recursivas

$\mathbb{N}$  es el conjunto más pequeño tal que:

- $0 \in \mathbb{N}$  (condición inicial)
- Si  $x \in \mathbb{N}$  entonces  $sx \in \mathbb{N}$  (condición iterativa)



# Números Naturales

## Definiciones Recursivas

$\mathbb{N}$  es el conjunto más pequeño tal que:

- $0 \in \mathbb{N}$  (condición inicial)
- Si  $x \in \mathbb{N}$  entonces  $sx \in \mathbb{N}$  (condición iterativa)
- La definición inductiva de  $\mathbb{N}$  permite definir funciones recursivamente:

$$sum(x, 0) = x \quad sum(x, sy) = s(sum(x, y))$$



# Listas Finitas

## Definiciones Recursivas

$\mathcal{L}(A)$  es el conjunto más pequeño tal que:



# Listas Finitas

## Definiciones Recursivas

$\mathcal{L}(A)$  es el conjunto más pequeño tal que:

- $nil \in \mathcal{L}(A)$ .



# Listas Finitas

## Definiciones Recursivas

$\mathcal{L}(A)$  es el conjunto más pequeño tal que:

- $nil \in \mathcal{L}(A)$ .
- Si  $a \in A$  y  $\ell \in \mathcal{L}(A)$  entonces  $cons(a, \ell) \in \mathcal{L}(A)$ .



# Listas Finitas

## Definiciones Recursivas

$\mathcal{L}(A)$  es el conjunto más pequeño tal que:

- $nil \in \mathcal{L}(A)$ .
- Si  $a \in A$  y  $\ell \in \mathcal{L}(A)$  entonces  $cons(a, \ell) \in \mathcal{L}(A)$ .
- La función longitud  $len : \mathcal{A} \rightarrow \mathbb{N}$  se define recursivamente como:

$$len(nil) = 0 \quad len(cons(a, \ell)) = len(\ell) + 1$$



# Paradigma de Modelación Inductiva

## Inducción

De los ejemplos anteriores podemos concluir:

- La inducción es un proceso de definición y razonamiento que determina conjuntos (tipos de datos) **numerables** cuyos elementos son **estructuras finitas** construibles a partir de ciertos **objetos básicos**.



# Paradigma de Modelación Inductiva

## Inducción

De los ejemplos anteriores podemos concluir:

- La inducción es un proceso de definición y razonamiento que determina conjuntos (tipos de datos) **numerables** cuyos elementos son **estructuras finitas** construibles a partir de ciertos **objetos básicos**.
- La inducción permite **definir funciones**  $f : \mathcal{I} \rightarrow A$  mediante principios de recursión, definiendo el valor de  $f$  en cada **constructor** de  $I$ .



# Paradigma de Modelación Inductiva

## Inducción

De los ejemplos anteriores podemos concluir:

- La inducción es un proceso de definición y razonamiento que determina conjuntos (tipos de datos) **numerables** cuyos elementos son **estructuras finitas** construibles a partir de ciertos **objetos básicos**.
- La inducción permite **definir funciones**  $f : \mathcal{I} \rightarrow A$  mediante principios de recursión, definiendo el valor de  $f$  en cada **constructor** de  $I$ .



# Paradigma de Modelación Inductiva

## Inducción

De los ejemplos anteriores podemos concluir:

- La inducción es un proceso de definición y razonamiento que determina conjuntos (tipos de datos) **numerables** cuyos elementos son **estructuras finitas** construibles a partir de ciertos **objetos básicos**.
- La inducción permite **definir funciones**  $f : \mathcal{I} \rightarrow A$  mediante principios de recursión, definiendo el valor de  $f$  en cada **constructor** de  $I$ . En el caso de  $\mathcal{I} = \mathbb{N}$  mediante ecuaciones de la forma:

$$f(0) = a \quad f(s(n)) = g(f(n))$$

con  $a \in A$  y  $g : A \rightarrow A$ .



# Principios de Inducción

Cualquier estructura definida recursivamente genera un principio de inducción útil como herramienta de demostración.



# Principios de Inducción

Cualquier estructura definida recursivamente genera un principio de inducción útil como herramienta de demostración.

- El principio de inducción para naturales: si  $0 \in P$  y si para cada  $x \in P$  se verifica también  $sx \in P$  entonces  $\mathbb{N} \subseteq P$ .



# Principios de Inducción

Cualquier estructura definida recursivamente genera un principio de inducción útil como herramienta de demostración.

- El principio de inducción para naturales: si  $0 \in P$  y si para cada  $x \in P$  se verifica también  $sx \in P$  entonces  $\mathbb{N} \subseteq P$ .
- El principio de inducción para listas: si  $nil \in P$  y si para cada  $a \in A$  y  $x \in P$  se verifica también  $cons(a, x) \in P$  entonces  $(A) \subseteq P$ .



## Definición recursiva de $\Sigma^*$

Dado un alfabeto  $\Sigma$ , el conjunto  $\Sigma^*$  que consta de todas las cadenas de símbolos de  $\Sigma$  puede definirse recursivamente como sigue:

- $\varepsilon \in \Sigma^*$ .



## Definición recursiva de $\Sigma^*$

Dado un alfabeto  $\Sigma$ , el conjunto  $\Sigma^*$  que consta de todas las cadenas de símbolos de  $\Sigma$  puede definirse recursivamente como sigue:

- $\varepsilon \in \Sigma^*$ .
- Si  $v \in \Sigma^*$  y  $a \in \Sigma$  entonces  $va \in \Sigma^*$



## Definición recursiva de $\Sigma^*$

Dado un alfabeto  $\Sigma$ , el conjunto  $\Sigma^*$  que consta de todas las cadenas de símbolos de  $\Sigma$  puede definirse recursivamente como sigue:

- $\varepsilon \in \Sigma^*$ .
- Si  $v \in \Sigma^*$  y  $a \in \Sigma$  entonces  $va \in \Sigma$
- Son todas.



# Principio de Inducción para $\Sigma^*$

Sea  $P \subseteq \Sigma^*$  un conjunto de cadenas.



# Principio de Inducción para $\Sigma^*$

Sea  $P \subseteq \Sigma^*$  un conjunto de cadenas. Si

1  $\varepsilon \in P$  y



# Principio de Inducción para $\Sigma^*$

Sea  $P \subseteq \Sigma^*$  un conjunto de cadenas. Si

- 1  $\varepsilon \in P$  y
- 2 Si para cualquier  $a \in \Sigma$  y  $v \in P$  se verifica que  $va \in P$



# Principio de Inducción para $\Sigma^*$

Sea  $P \subseteq \Sigma^*$  un conjunto de cadenas. Si

①  $\varepsilon \in P$  y

② Si para cualquier  $a \in \Sigma$  y  $v \in P$  se verifica que  $va \in P$

entonces cualquier  $u \in \Sigma^*$  pertenece a  $P$ . Es decir,  $\Sigma^* = P$ .



# Definición Recursiva de Concatenación

Formalmente la concatenación de cadenas es una función

$$\cdot : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$



# Definición Recursiva de Concatenación

Formalmente la concatenación de cadenas es una función

$$\cdot : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

definida por:

- $u \cdot \varepsilon = u$



# Definición Recursiva de Concatenación

Formalmente la concatenación de cadenas es una función

$$\cdot : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

definida por:

- $u \cdot \varepsilon = u$
- $u \cdot (va) = (u \cdot v)a$



# Definición Recursiva de Concatenación

Formalmente la concatenación de cadenas es una función

$$\cdot : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

definida por:

- $u \cdot \varepsilon = u$
- $u \cdot (va) = (u \cdot v)a$



# Definición Recursiva de Concatenación

Formalmente la concatenación de cadenas es una función

$$\cdot : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

definida por:

- $u \cdot \varepsilon = u$
- $u \cdot (va) = (u \cdot v)a$

**Pizarrón:** probemos que la concatenación es asociativa.



# Longitud y reversa

## Definiciones recursivas

- Longitud:  $|\cdot| : \Sigma^* \rightarrow \mathbb{N}$

$$|\varepsilon| = 0 \qquad |va| = |v| + 1$$



# Longitud y reversa

## Definiciones recursivas

- Longitud:  $|\cdot| : \Sigma^* \rightarrow \mathbb{N}$

$$|\varepsilon| = 0 \qquad |va| = |v| + 1$$

- Reversa:  $\cdot^R : \Sigma^* \rightarrow \Sigma^*$

$$\varepsilon^R = \varepsilon \qquad (va)^R = av^R$$

