Introducción a la Teoría de Autómatas, Lenguajes y Computación

Gustavo Rodríguez Gómez y Aurelio López López

INAOE

Propedéutico 2020

Capítulo 1
Introducción
Autómatas

Libro de texto

- John E. Hopcroft et al., "Introduction to Automata Theory, Languajes, and Computation", segunda edición, Addison Wesley

Motivación

- La teoría de autómatas es el estudio de dispositivos (*máquinas*) de computación abstractas.

Motivación

- La teoría de autómatas es el estudio de dispositivos (*máquinas*) de computación abstractas.
- Alan Turing estudio máquinas abstractas, las máquinas de Turing, antes que existieran las computadoras reales.

Motivación

- La teoría de autómatas es el estudio de dispositivos (*máquinas*) de computación abstractas.
- Alan Turing estudio máquinas abstractas, las máquinas de Turing, antes que existieran las computadoras reales.
- Objetivo de Turing en 1930:

Motivación

- La teoría de autómatas es el estudio de dispositivos (*máquinas*) de computación abstractas.
- Alan Turing estudio máquinas abstractas, las máquinas de Turing, antes que existieran las computadoras reales.
- Objetivo de Turing en 1930:
  - Describir en forma precisa la frontera entre lo que una *computadora* puede hacer y lo que no puede hacer.

Motivación

- La teoría de autómatas es el estudio de dispositivos (*máquinas*) de computación abstractas.
- Alan Turing estudio máquinas abstractas, las máquinas de Turing, antes que existieran las computadoras reales.
- Objetivo de Turing en 1930:
  - Describir en forma precisa la frontera entre lo que una *computadora* puede hacer y lo que no puede hacer.
- Entre 1940 y 1950 surgen las máquinas hoy llamadas "autómatas finitos".

Motivación

- La teoría de autómatas es el estudio de dispositivos (*máquinas*) de computación abstractas.
- Alan Turing estudio máquinas abstractas, las máquinas de Turing, antes que existieran las computadoras reales.
- Objetivo de Turing en 1930:
  - Describir en forma precisa la frontera entre lo que una *computadora* puede hacer y lo que no puede hacer.
- Entre 1940 y 1950 surgen las máquinas hoy llamadas "autómatas finitos".
- A finales de 1950 el lingüista Chomsky inicia el estudio de las "*gramáticas*" formales.

Motivación

- En 1969 S. Cook pudo clasificar los problemas que pueden ser resueltos en una computadora en dos categorías:

Motivación

- En 1969 S. Cook pudo clasificar los problemas que pueden ser resueltos en una computadora en dos categorías:
  - problemas que se pueden resolver en forma eficiente,

Motivación

- En 1969 S. Cook pudo clasificar los problemas que pueden ser resueltos en una computadora en dos categorías:
  - problemas que se pueden resolver en forma eficiente,
  - problemas que en principio se pueden resolver pero que en la práctica consumen mucho tiempo (NP–duros, intratables).

Motivación

- En 1969 S. Cook pudo clasificar los problemas que pueden ser resueltos en una computadora en dos categorías:
  - problemas que se pueden resolver en forma eficiente,
  - problemas que en principio se pueden resolver pero que en la práctica consumen mucho tiempo (NP–duros, intratables).
- Todos los desarrollos teóricos se apoyan en lo que los científicos de la computación desarrollan actualmente.

Motivación

- Los autómatas finitos y las gramáticas formales se usan como modelos para

Motivación

- Los autómatas finitos y las gramáticas formales se usan como modelos para
  - Construir el software para el diseño de circuitos digitales

Motivación

- Los autómatas finitos y las gramáticas formales se usan como modelos para
  - Construir el software para el diseño de circuitos digitales
  - Construir el analizador léxico de un compilador

Motivación

- Los autómatas finitos y las gramáticas formales se usan como modelos para
  - Construir el software para el diseño de circuitos digitales
  - Construir el analizador léxico de un compilador
- La máquinas de Turing nos ayuda a entender lo que podemos esperar de nuestro software.

Motivación

- Los autómatas finitos y las gramáticas formales se usan como modelos para
  - Construir el software para el diseño de circuitos digitales
  - Construir el analizador léxico de un compilador
- La máquinas de Turing nos ayuda a entender lo que podemos esperar de nuestro software.
- La teoría de problemas intratables nos ayuda a deducir si nos enfrentamos con problemas tratables o no.

Aplicaciones de los Autómatas Finitos

- Algunas aplicaciones de autómatas finitos

Aplicaciones de los Autómatas Finitos

- Algunas aplicaciones de autómatas finitos
  - Diseño de software y verificación del comportamiento de circuitos digitales.

Aplicaciones de los Autómatas Finitos

- Algunas aplicaciones de autómatas finitos
  - Diseño de software y verificación del comportamiento de circuitos digitales.
  - Analizadores léxicos de compiladores.

Aplicaciones de los Autómatas Finitos

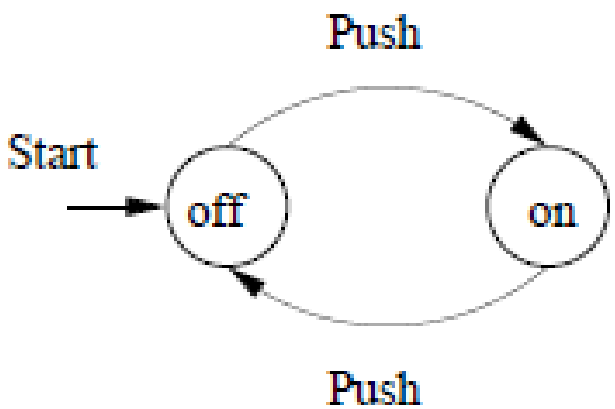
- Algunas aplicaciones de autómatas finitos
  - Diseño de software y verificación del comportamiento de circuitos digitales.
  - Analizadores léxicos de compiladores.
  - Software para explorar grandes volúmenes de texto y encontrar patrones.

Aplicaciones de los Autómatas Finitos

- Algunas aplicaciones de autómatas finitos
  - Diseño de software y verificación del comportamiento de circuitos digitales.
  - Analizadores léxicos de compiladores.
  - Software para explorar grandes volúmenes de texto y encontrar patrones.
  - Software para verificar sistemas que tengan un número finito de estados distintos, por ejemplo, protocolos de comunicación.

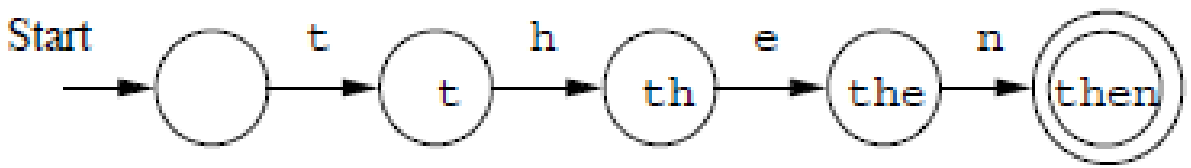
Ejemplo de Autómata Finito 1

- Autómata finito que modela un interruptor encendido/apagado (on/off)



Ejemplo de Autómata Finito 2

- Autómata finito para reconocer la cadena "then"



Gramáticas y Expresiones Regulares

Existen dos formas alternativas de especificar una máquina

1. **Gramáticas:** una regla gramatical como $E \Longrightarrow E + E$ establece una expresión aritmética
2. **Expresiones Regulares:** denota una estructura de datos '$[A - Z]\,[a - z] * []\,[A - Z]\,[A - Z]$' que representa un patrón en un texto

Condiciones Suficientes

Una proposición A es una condición suficiente de una proposición B si

$$A \Rightarrow B \qquad (\sim (A \Rightarrow B) = \sim B \Rightarrow \sim A)$$

Examples (**Suficiencia**)

Si llueve mi patio se moja $(A \Rightarrow B)$.Es suficiente que llueva para que mi patio se moje, pero no es necesario que llueva para que mi patio se moje.

Examples (**Suficiencia**)

Un elefante es un animal $(A \Rightarrow B)$. Es suficiente ser elefante para ser un animal, pero no es necesario ser elefante para ser un animal.

Condiciones Necesarias

Una proposición B es una condición necesaria de una proposición A si

$$A \Leftarrow B$$

Example (Necesidad)

Nací en México $\not\Rightarrow$ nací en Hermosillo. Sin embargo, si nací en Hermosillo $\Rightarrow$ nací en México ($A \Leftarrow B$).

Condición Suficiente y Necesaria

Las proposiciones A y B son suficiente y necesaria si se cumple

$$A \Leftrightarrow B$$

Example (Suficiente y Necesaria)

Si n es un entero par $\Rightarrow$ n divisible por 2. Si n entero es divisible por 2 $\Rightarrow$ n es entero par. En este caso se cumple $A \Leftrightarrow B$.

Tipos de Demostraciones

1. Deductivas
2. Relacionados a conjuntos
3. Contrapositiva
4. Por contradicción
5. Por contrajemplos
6. Inductivas (objetos definidos recursivamente)

Alfabetos

- Un **alfabeto**

$$\Sigma = \text{ conjunto de símbolos } \textbf{finito} \text{ no vacío}$$

Alfabetos

- Un **alfabeto**

$$\Sigma = \text{ conjunto de símbolos } \textbf{finito} \text{ no vacío}$$

- Ejemplos

$\Sigma = \{0, 1\}$, el alfabeto binario,

$\Sigma = \{a, b, c, \ldots, z\}$, conjunto de todas las letras minúsculas

$\Sigma =$ conjunto de todos los caracteres ASCII

Cadenas

- Una **cadena** w es una sucesión finita de símbolos de algún alfabeto Σ

$$w = a_1 a_2 \cdots a_i, \quad a_k \in \Sigma \quad k = 1, 2, \ldots, i$$

Cadenas

- Una **cadena** w es una sucesión finita de símbolos de algún alfabeto Σ

$$w = a_1 a_2 \cdots a_i, \quad a_k \in \Sigma \quad k = 1, 2, \ldots, i$$

- Ejemplo, la cadena 01011 es una cadena del alfabeto $\Sigma = \{0, 1\}$

Cadenas

- Una **cadena** w es una sucesión finita de símbolos de algún alfabeto Σ

$$w = a_1 a_2 \cdots a_i, \quad a_k \in \Sigma \quad k = 1, 2, \ldots, i$$

- Ejemplo, la cadena 01011 es una cadena del alfabeto $\Sigma = \{0, 1\}$
- La **cadena vacía** ϵ es la cadena con cero ocurrencias de símbolos de Σ.

Cadenas

- Una **cadena** w es una sucesión finita de símbolos de algún alfabeto Σ

$$w = a_1 a_2 \cdots a_i, \quad a_k \in \Sigma \quad k = 1, 2, \ldots, i$$

- Ejemplo, la cadena 01011 es una cadena del alfabeto $\Sigma = \{0, 1\}$
- La **cadena vacía** ϵ es la cadena con cero ocurrencias de símbolos de Σ.
- La longitud de una cadena $w = a_1 a_2 \cdots a_i$ es el número de posiciones de los símbolos de la cadena

$$|w| = i$$

Cadenas

- Una **cadena** w es una sucesión finita de símbolos de algún alfabeto Σ

$$w = a_1 a_2 \cdots a_i, \quad a_k \in \Sigma \quad k = 1, 2, \ldots, i$$

- Ejemplo, la cadena 01011 es una cadena del alfabeto $\Sigma = \{0, 1\}$
- La **cadena vacía** ϵ es la cadena con cero ocurrencias de símbolos de Σ.
- La longitud de una cadena $w = a_1 a_2 \cdots a_i$ es el número de posiciones de los símbolos de la cadena

$$|w| = i$$

- Ejemplo

$$|abcde| = 5, \quad |\epsilon| = 0$$

Potencias de un alfabeto

- Sea Σ un alfabeto, definimos las **potencias** de Σ por

$$\Sigma^k = \{w | w \text{ es una cadena de } \Sigma \text{ y } |w| = k\}, \quad k \geq 0$$

Potencias de un alfabeto

- Sea Σ un alfabeto, definimos las **potencias** de Σ por

$$\Sigma^k = \{w | w \text{ es una cadena de } \Sigma \text{ y } |w| = k\}, \quad k \geq 0$$

- $\Sigma^0 = \{\epsilon\}$ para cualquier alfabeto

Potencias de un alfabeto

- Sea Σ un alfabeto, definimos las **potencias** de Σ por

$$\Sigma^k = \{w | w \text{ es una cadena de } \Sigma \text{ y } |w| = k\}, \quad k \geq 0$$

- $\Sigma^0 = \{\epsilon\}$ para cualquier alfabeto
- Si $\Sigma = \{0, 1\}$ entonces

$$\Sigma^1 = \{0, 1\},$$
$$\Sigma^2 = \{00, 01, 10, 11\},$$
$$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

Preguntas

Preguntas

1. ¿Cuántas cadenas hay en Σ^4?
2. ¿Cuál es la diferencia entre Σ y Σ^1?

Conjunto de todas las cadenas de un alfabeto

- El conjunto de todas las cadenas de un alfabeto Σ es denotado por Σ^*

$$\begin{aligned}\Sigma^* &= \{w | w \text{ es cadena de } \Sigma\}, \\ &= \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots\end{aligned}$$

Conjunto de todas las cadenas de un alfabeto

- El conjunto de todas las cadenas de un alfabeto Σ es denotado por Σ^*

$$\begin{aligned}\Sigma^* &= \{w | w \text{ es cadena de } \Sigma\}, \\ &= \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots\end{aligned}$$

- Ejemplo

$$\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \cdots\}$$

Cadenas no vacías

- El conjunto de **cadenas no vacías de un alfabeto** se define por

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$$

en consecuencia

$$\Sigma^* = \Sigma^+ \cup \{\epsilon\}$$

Concatenación de cadenas

- Sean $x = a_1 a_2 \cdots a_i$, $y = b_1 b_2 \cdots b_j$ cadenas de Σ, definimos la **concatenación** de x con y por

$$xy = a_1 a_2 \cdots a_i b_1 b_2 \cdots b_j,$$

Concatenación de cadenas

- Sean $x = a_1 a_2 \cdots a_i$, $y = b_1 b_2 \cdots b_j$ cadenas de Σ, definimos la **concatenación** de x con y por

$$xy = a_1 a_2 \cdots a_i b_1 b_2 \cdots b_j,$$

- La longitud de la nueva cadena es

$$|xy| = i + j$$

Concatenación de cadenas

- Sean $x = a_1 a_2 \cdots a_i$, $y = b_1 b_2 \cdots b_j$ cadenas de Σ, definimos la **concatenación** de x con y por

$$xy = a_1 a_2 \cdots a_i b_1 b_2 \cdots b_j,$$

- La longitud de la nueva cadena es

$$|xy| = i + j$$

- Ejemplo $x = 01110$, $y = 1111 \Rightarrow xy = 011101111$, $|xy| = 9$.

Concatenación de cadenas

- Sean $x = a_1 a_2 \cdots a_i$, $y = b_1 b_2 \cdots b_j$ cadenas de Σ, definimos la **concatenación** de x con y por

$$xy = a_1 a_2 \cdots a_i b_1 b_2 \cdots b_j,$$

- La longitud de la nueva cadena es

$$|xy| = i + j$$

- Ejemplo $x = 01110$, $y = 1111 \Rightarrow xy = 011101111$, $|xy| = 9$.
- Para cualquier cadena x se cumple

$$x\epsilon = \epsilon x = x$$

Lenguajes

Definition

Sea Σ un alfabeto y $L \subseteq \Sigma^*$ diremos entonces que L es un lenguaje de Σ.

Examples

Ejemplos de lenguajes

1. El conjunto de palabras (aceptadas) del español
2. El conjunto de todos los programas (correctos) de C
3. El el lenguaje de todas las cadenas que consisten de n 0's seguidos por n 1's para alguna $n \geq 0$

$$\{\epsilon, 01, 0011, 000111, \ldots\}$$

Más Ejemplos de Lenguajes

Examples

Ejemplos de lenguajes

1. El lenguaje de todas las cadenas 0's y n 1's con igual número de cada uno

$$\{\epsilon, 01, 01, 0011, 0101, 1001, \ldots\}$$

2. El lenguaje formado por los números binarios cuyo valor es un primo

$$L_p = \{10, 11, 101, 111, 1011\}$$

3. Σ^* es un lenguaje para cualquier alfabeto Σ.

Más Ejemplos de Lenguajes

Examples

1. ϕ, el lenguaje vacío, es un lenguaje de cualquier alfabeto Σ.
2. $\{\epsilon\}$, el lenguaje que consiste únicamente de la cadena vacía, es un lenguaje de cualquier alfabeto Σ.
3. Observación $\phi \neq \{\epsilon\}$.

Problemas en teoría de autómatas

Definition

Sea Σ un alfabeto y L un lenguaje de Σ.El problema L es:

- Dado una cadena $w \in \Sigma^*$ decidir si $w \in L$ ó $w \notin L$.

Examples

Dado una cadena w de 0's y 1's averiguar si $w \in L_p$ o $w \notin L_p$