Autómatas y Lenguajes Formales 2019-II
Facultad de Ciencias UNAM*
Nota de Clase 1: Introducción, cadenas y lenguajes

Favio E. Miranda Perea A. Liliana Reyes Cabello
Lourdes del Carmen González Huesca

25 de enero de 2019

1. Fundamentos de la computación

Las ciencias de la computación como un conglomerado de disciplinas científicas y de ingeniería relacionadas con el estudio y aplicación del cómputo van desde las más puras y básicas dedicadas a los fundamentos de la computación, hasta las ingenierías dedicadas a aplicaciones especificas.

Los fundamentos de la computación se dividen esencialmente en dos partes:

- **Teoría de la Programación:**
  Dedicada a estudiar los lenguajes de programación, los cuales nos sirven para implementar procesos de cómputo.
- **Teoría de la Computación:** Dedicada a entender la naturaleza del cómputo, sus posibilidades y limitaciones.

Teoría de la Programación Se divide en diversas disciplinas, como son:

- Lógica Computacional.
- Teoría de lenguajes de programación: estilos (lógico, funcional, imperativo, orientado a objetos, etc), semántica operacional, tipado, etc.
- Métodos formales: especificación y verificación

Teoría de la Computación Algunas de las disciplinas que conforman esta área son:

- Algoritmos y estructuras de datos
- Complejidad computacional
- Cómputo paralelo y distribuido

*Material elaborado en el marco del proyecto PAPIME PE102117

- Teoría de autómatas

Mediante la teoría del cómputo, especificamente mediante la teoría de autómatas y lenguajes formales, podemos tratar preguntas como:

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

Al responder a estas preguntas, la teoría de la computación nos permitirá entender cuáles son las capacidades y limitaciones fundamentales de una computadora. Las nociones de *cómputo* y *computabilidad efectiva* serán capturadas mediante abstracción matemática y la modelación de autómatas.

2. Abstracción del concepto de computadora

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/codominio de tal función? es decir, qué tipos de datos se esperan como entrada: números, palabras, textos, imágenes, etc.
- ¿ Cómo podemos representar los datos de entrada de manera uniforme?

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?

3. Conceptos Fundamentales

Los tres conceptos fundamentales a estudiar en autómatas y lenguajes formales son:

- Lenguajes: conjuntos de cadenas.
- Gramáticas: mecanismos para generar cadenas.

- Autómatas: mecanismos para procesar cadenas.

Nuestro objetivo principal es estudiar las relaciones entre estos tres conceptos para responder a las preguntas hechas al principio de esta nota.

Primero daremos una breve introducción a las gramáticas y los autómatas así como un poco de sus orígenes y aplicaciones. Después continuaremos con una teoría de cadenas para seguir con los lenguajes. El estudio a fondo de los autómatas será el resto del curso.

3.1. Gramáticas

Un mecanismo relevante para generar un lenguaje es mediante el concepto de gramática formal.

Las gramáticas formales fueron introducidas por Chomsky en 1956 cuya intención era tener un modelo para la descripción de lenguajes naturales. Posteriormente se utilizaron como herramienta para presentar la sintaxis de lenguajes de programación y para el diseño de analizadores léxicos de compiladores.

3.2. Autómatas

- Un autómata es una representación abstracta de una máquina
- Los aspectos relevantes para nosotros son el diseño y la especificación de autómatas.
- La abstracción captura únicamente el comportamiento de una máquina, es decir, las secuencias de eventos que ocurren.
- Estas máquinas abstractas nos serán útiles en el procesamiento y análisis de lenguajes, al procesar cadenas *aceptándolas o rechazándolas.*

3.2.1. Un poco de historia

- En su artículo "On Computable Numbers, with an Application to the Entscheidungsproblem" de 1936, Turing reformula los resultados de Kurt Gödel de 1931 acerca de las limitaciones de pruebas y cómputos. Reemplaza el lenguaje formal universal de Gödel con lo que hoy llamamos máquinas de Turing que son dispositivos formales y simples de una computadora. Turing demostró que tales máquinas podrían der capaces de llevar a cabo cualquier cómputo matemático concebible, si este se representaba mediante un algoritmo.
- En el artículo "A logical calculus of the ideas immanent in nervous activity" que apareció en el Bull. Math. Biophysics 5 (1943), pp. 115-133, el neurofisíologo Warren McCulloch y el lógico Walter Pitts, desarrollaron modelos de redes neuronales con ciclos de retroalimentación basados en su visión de la neurología. Este fue un intento para modelar la estructura de los nervios.
- El modelo de Pitts-McCulloch fue simplificado por el lógico Stephen C. Kleene en 1956, en el que se considera el primer artículo acerca de autómatas finitos y expresiones regulares, "Representation of events in nerve nets and finite automata".

- En 1959 Rabin y Scott presentan una máquina con un número finito de estados que simplifica a la máquina de Turing. Por su artículo "Finite Automata and Their Decision Problem", que introduce la idea de máquinas no deterministicas, un concepto invaluable, ambos reciben el premio Turing en 1976.
- En 1968 Ken Thompson usa la noción de expresión regular definida por Kleene en el sistema UNIX.

3.2.2. Aplicaciones

Los autómatas tienen diversas aplicaciones en computación, por ejemplo:

- Software para hallar patrones en una gran cantidad de texto, por ejemplo en colecciones de páginas web.
- Diseño de circuitos digitales.
- Analizadores léxicos para compiladores.
- Búsqueda de palabras clave en internet.
- Verificación de sistemas con un número finito de estados (por ejemplo protocolos de comunicación).
- Verificación de modelos, modelado y verificación de sistemas empotrables o embebidos.
- Aplicaciones en genética, patrones regulares en proteinas.
- Aplicaciones en lingüistica, construcción de diccionarios grandes, correctores de ortografía.
- Modelado de máquinas reales: relojes, telefonos, seguros de auto, etc.
- Modelado de sistemas discretos en general.

Existen diversos tipos de autómatas, nosotros estudiaremos:

- Autómatas con un número finito de estados: determinísticos, no determinísticos.
- Autómatas de pila
- Autómatas Linealmente Acotados
- Máquinas de Turing.

Otras clases de autómatas que no estudiaremos aquí son los probabilísticos, celulares, paralelos, de árbol, etc.

4. Cadenas

Las cadenas son los objetos fundamentales para la mayoría de los sistemas formales. En esta sección revisamos las operaciones de importancia entre cadenas.

Los *símbolos* son los objetos más simples con los que trataremos, ellos conforman a las cadenas. Una definición formal de símbolo nos llevaría al ámbito de la filosofía del lenguaje, así que para nuestros propositos basta decir que:

Definición 1 *Un símbolo o caracter es una entidad considerada indivisible.*

Por comodidad utilizaremos como símbolos:

- a,b,c,d,e, ...
- 0,1,2,3, ..., 9

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

Definición 2 *Un alfabeto es un conjunto finito de símbolos.*

Por lo general usaremos las letras griegas Σ y Γ para denotar alfabetos.

Ejemplos:

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

Definición 3 *Una* cadena, expresión o palabra *es una sucesión* ***finita*** *de símbolos tomados de un alfabeto dado* Σ.

En otros ámbitos se permiten cadenas con un número infinito de símbolos pero nunca en nuestro curso.

Ejemplos:

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

Obsérvese que los símbolos son a su vez cadenas que constan de un solo caracter. Más aún podemos referirnos a la *cadena vacía* (i.e. la sucesión vacía de símbolos) que se denotará con el metasímbolo ε [1]. Este metasímbolo no es ni una palabra ni un símbolo. Es decir estará prohibido considerar a ε como símbolo de un alfabeto Σ.

[1] Algunos autores denotan a la cadena vacía con λ o Λ.

Definición 4 *Al conjunto infinito de todas las cadenas sobre un alfabeto dado Σ se le denota como $\Sigma^\star$ y es llamado la estrella o cerradura de Kleene*[2] *de Σ. Es decir $\Sigma^\star = \{a_1 \dots a_n \mid a_i \in \Sigma\}$ donde $a_1 \dots a_n$ es una cadena.*

4.1. Operaciones con cadenas

Las cadenas son objetos que se manipulan para estudiar los lenguajes, para ello se consideran algunas operaciones sobre cadenas:

Definición 5 *La longitud de una cadena w es el número de símbolos en w. Esta función se define como $|\cdot| : \Sigma^\star \to \mathbb{N}$. Si $w = a_1 \dots a_n$ entonces $|w| = n$.*

Definición 6 *La operación básica entre cadenas es la concatenación $\cdot : \Sigma^\star \times \Sigma^\star \to \Sigma^\star$. Consiste en unir cadenas en orden de izquierda a derecha: si v, w son cadenas entonces $v \cdot w$ será la cadena obtenida al pegar v con w. Usualmente el operador $\cdot$ no se usa explícitamente y escribimos simplemente uw en lugar de $u \cdot w$.*

Ejemplos:

- La concatenación de *cala* y *baza* es la cadena *calabaza*.
- Si $v = broco$ y $w = li$ entonces $vw = brocoli$.
- Si $x = champu$ y $y = rrado$ entonces $yx = rradochampu$.

Algunas propiedades de esta operación son las siguientes:

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

Además es claro que no es una operación conmutativa, por ejemplo si $w = ja$ y $v = rrito$ entonces $wv = jarrito \neq rritoja = vw$.

En particular podemos denotar la concatenación repetida de una misma cadena como sigue: $w = w^1$, $ww = w^2$, $\dots$, $\underbrace{w \dots w}_{\text{n-veces}} = w^n$ y en particular $w^0 = \varepsilon$.

Definición 7 *La reversa de una cadena u, denotada u^R, contiene los símbolos del último al primero, es decir si $u = a_1 a_2 \dots a_n$ entonces $u^R = a_n a_{n-1} \dots a_2 a_1$.*

Además cumple las siguientes propiedades:

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

[2] Stephen Cole Kleene, 1909–1994, prominente lógico matemático.

4.2. Subcadenas, prefijos y sufijos

Las cadenas pueden subdividirse para su estudio:

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

5. Recursión

La recursión como mecanismo de definición de tipos de datos, de definición de funciones y como una herramienta fundamental en programación será de gran utilidad.

5.1. Definiciones recursivas: recordatorio

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

- $0 \in \mathbb{N}$
- Si $x \in \mathbb{N}$ entonces $sx \in \mathbb{N}$

Esta definición de $\mathbb{N}$ permite definir funciones recursivamente, por ejemplo la suma de números naturales

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

Listas de objetos Dado una coleccion A, $\mathcal{L}(A)$ es el conjunto más pequeño tal que:

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

Así también podemos definir funciones recursivas como por ejemplo la función que calcula la longitud de una lista $len : \mathcal{A} \to \mathbb{N}$ dada por:

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

5.2. Modelación Inductiva

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 del estilo $f : \mathcal{I} \to A$ mediante principios de recursión, definiendo el valor de f en cada constructor de I.
  Por ejemplo en el caso de $\mathcal{I} = \mathbb{N}$, sólo es necesario dar ecuaciones de la forma:

$$(0) = a \qquad f(s(n)) = g(f(n)) \quad \text{donde } a \in A,\ g : A \to A$$

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 tambien $sx \in P$ 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 tambien $cons(a, x) \in P$ entonces $\mathcal{L}(A) \subseteq P$.

5.3. Inducción en cadenas

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

- $\varepsilon \in \Sigma^\star$.
- Si $w \in \Sigma^\star$ y $a \in \Sigma$ entonces $wa \in \Sigma^\star$ [3]
- Son todas.

Esta definición recursiva genera el siguiente principio de inducción estructural:

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

1. $\varepsilon \in P$ y
2. si para cualquier $a \in \Sigma$ y $w \in P$ se verifica que $wa \in P$

Entonces cualquier $u \in \Sigma^\star$ pertenece a P. Es decir, $\Sigma^\star = P$.

Siguiendo la estructura anterior, se definen las funciones anteriores recursivamente como sigue:

- La concatenación de cadenas es una función $\cdot : \Sigma^\star \times \Sigma^\star \to \Sigma^\star$ tal que:

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

- La longitud $|\cdot| : \Sigma^\star \to \mathbb{N}$ se define como:

$$|\boldsymbol{\varepsilon}| = 0 \qquad |va| = |v| + 1$$

- La reversa de una cadena ${}^{\cdot R} : \Sigma^\star \to \Sigma^\star$ está definida por:

$$\boldsymbol{\varepsilon}^R = \boldsymbol{\varepsilon} \qquad (va)^R = av^R$$

[3] Opcionalmente se puede dar una definción equivalente en donde los símbolos se agregen al inicio de una cadena: si $w \in \Sigma^\star$ y $a \in \Sigma$ entonces $aw \in \Sigma^\star$. El la literatura es más usada la convención que aparece en nuestra definción.

6. Lenguajes

Definición 8 *Un lenguaje $\mathcal{L}$ sobre un alfabeto Σ es simplemente un conjunto de cadenas de Σ, es decir $\mathcal{L} \subseteq \Sigma^\star$.*

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

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

Obsérvese que un lenguaje puede ser finito o infinito, que $\varnothing$ es un lenguaje, llamado *lenguaje vacío* y que $\Sigma^\star$ es un lenguaje llamado el *lenguaje total*. Además Σ y $\{\varepsilon\}$ también son lenguajes.

6.1. Operaciones con Lenguajes

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

$$L \cup M \qquad L \cap M \qquad L - M \qquad \bar{L} = \Sigma^\star - L$$

tambien son lenguajes. Adicionalmente tenemos las siguientes operaciones relevantes:

6.1.1. Concatenación de Lenguajes

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

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

Y cumple las siguientes propiedades:

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

También consideramos la concatenación repetida de un mismo lenguaje llamada *potencia* de un lenguaje como:

$$L^n = \underbrace{L \ldots L}_{\text{n-veces}}$$

6.1.2. Reversa de un Lenguaje

La reversa de un lenguaje se define como

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

Y cumple las siguientes propiedades:

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

6.1.3. Cerradura de Kleene

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

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

donde L^i es la potencia i-ésima de L.

6.1.4. Cerradura Positiva

La cerradura positiva de un lenguaje se define como

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

6.1.5. Propiedades de los operadores de cerradura

Se cumplen las siguientes propiedades

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

7. 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 describibles finitamente?