Autómatas y lenguajes formales

Víctor Mijangos de la Cruz

vmijangosc @ ciencias.unam.com

I. Introducción a los autómatas y lenguajes formales

Información del curso

Nombre	Autómatas y lenguajes formales
Créditos	10
Total de horas	96

Objetivo del curso

Este curso es una introducción a los conceptos de autómatas, y a la teoría del lenguajes formales. Se tratarán los diferentes tipos de lenguajes, sus gramáticas y los autómatas asociados. Asimismo, se expondrán las propiedades de éstos y las capacidades de estos modelos para trabajar problemas de cómputo.

Temario

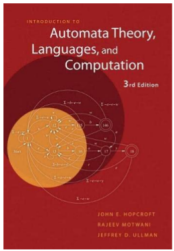
1. Introducción
   1. Motivación
   2. Marco histórico
   3. Introducción a los lenguajes formales
2. Lenguajes regulares
   1. Expresiones regulares
   2. Definición de lenguaje regular
   3. Propiedades de lenguajes regulares
3. Autómatas finitos
   1. Determinista y no-determinista
   2. Pumping lema
   3. Transformación entre representaciones
   4. Equivalencia y minimización
4. Lenguajes libres de contexto
   1. Propiedades de los LLC
   2. Formas normales
5. Autómatas con pila
   1. Lema de Ogden
   2. Decibilidad
6. Máquinas de Turing
   1. Máquina de Turing Universal y tesis Church-Turing
   2. Jerarquía de Chomsky
7. Introducción a la decibilidad
   1. Problema de detención
   2. Reducción
   3. Funciones computables

Bibliografía

Viso G., E. (2008), Introducción a la teoría de la computación. México: Las Prensas de Ciencias.

Ullman, J., Motwani, R. y Hopcroft, J. (2006). Introduction to Automata Theory, Languages, and Computation. Prentice Hall.

Rich, E. (2019). Automata, computability and complexity: Theory and Applications. Pearson Education.



Evaluación

La evaluación se realizará a partir de tres tareas y tres exámenes. El procentaje se dividirá de la siguiente forma:

Concepto	Porcentaje
Exámenes	40 %
Tareas	60 %

Notas: 1) Las tareas entregadas con retardo se penalizarán con 1 punto menos por cada día de retraso; 2) Los exámenes se harán en horario de clase con cámara abierta.

Motivación y definición

Motivación

La **teoría de la computación** es relevante para dar un fundamento matemático de la computación. Esto posibilita un mejor entendimiento de lo que posibilita sus aplicaciones prácticas

Pregunta de la teoría de la computación

Podemos entender que la pregunta a la que responde la teoría de la computación es la siguiente:

¿Cuáles son las capacidades y limitaciones fundamentales de las computadoreas?

Computadora

Una **computadora** es una máquina digital que puede ser programada para resolver diferentes tareas.

En la tarea de la computación nos interesan los métodos, procedimientos y, de forma general, los **algoritmos** que nos llevan a resolver problemas.

Asimismo, a la teoría de la computación le interesa conocer si estos algoritmos pueden ser implementados **eficientemente**.

De forma más general, podemos entender a una computadora como una **función**, que toma algunos argumentos y arroja un resultado.
Sin embargo, cabe preguntarse si toda función es **computable**; i.e., programable en un sistema digital.

Problemas en teoría de la computación

Son dos los problemas básicos al que nos enfrentamos en la teoría de la computación:

Decibilidad: Determina lo que una computadora puede hacer. Un problema será decidible si una computadora puede resolverlos.

Intractabilidad: Determina qué problemas pueden ser resuleto por una computadora de manera **eficiente**. Los problemas que se pueden resolver 'fácilmente' se denominan *tratables*.

Noción de decibilidad

Sistema formal

Un sistema formal para una teoría T consiste de un lenguaje $\mathcal{L}_T$ con los componentes:

- Fórmulas ϕ pertenecientes al lenguaje $\mathcal{L}_T$.
- Axiomas: un subconjunto de fórmulas del lenguaje $\mathcal{L}_T$ que se asumen como válidas.
- Reglas de inferencia que permiten derivar nuevas fórmulas o teoremas.

Teorema de incompletud de Gödel

Dado una teoría T sobre la aritmética de Peano con un sistema formal $\mathcal{L}_T$, si T es consistente (no contiene contradicciones), entonces T es indecidible; i.e., $\exists \phi \in \mathcal{L}_T \Big(T \not\models \phi \wedge T \not\models \neg\phi \Big)$

Marco histórico

- 1931. Gödel propone funciones recursivas primitivas.
- 1936 Kleeme extiende la idea de Gödel a funciones recursivas parciales.
- 1936 Church propone el cálculo Lambda.
- 1936 Turing estudia máquinas abastractas y propone la máquina de Turing.
- 1940-1950 Comienzan los estudios de Autómatas Finitos.
- 1954 Cadenas de Márkov como estudio probabilístico de cadenas.
- 1958 Chomsky estudia las gramáticas formales y propne la jerarquía de Chomsky
- 1959 Newell define los autómatas con pila.
- 1969 Cook extiende los estudios de Turing a los estudios de computabilidad/tratabilidad.

Tesis de Turing

Surgen **teoría de la computación** y el **test de Turing** en los 40s.

Tesis de Turing

Para cualquier sistema forma determinista, existe una máquina de Turing formalmente equivalente.

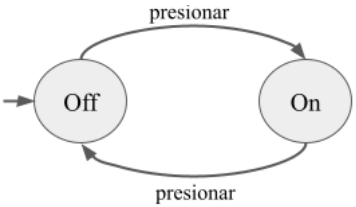
Máquinas universales

Las máquinas de Turing universales pueden programarse para imitar a cualquier máquina de Turing específica.

Noción de autómata

La palabra **autómata** viene del griego que se refiere a "aquello que actúa por sí mismo". En la teoría de la computación, se buscará estudiar **máquinas** de manera **abstracta**.

Un ejemplo de autómata puede ser un botón que al presionarlo enciende o apaga la máquina.



Algunas representaciones estructurales de los autómatas son:

1. Gramáticas
2. Expresiones regulares

Alfabetos, cadenas y lenguajes

Alfabeto

Para tratar los problemas de computación, usaremos como entradas elementos que conoceremos como cadenas. Para esto, necesitamos definiciones previas:

Símbolo

Un símbolo es un objeto indivisible que representa algo (sonido, concepto, entidad,...). Denotamos los símbolos con minúsculas a, b.

Un ejemplo de símbolos son los caracteres y los dígitos.

Alfabeto

Un alfabeto es un conjunto no vacío de símbolos (generalmente finito), al que denotamos como:

$$\Sigma = \{a_1, a_2, ..., a_n\}$$

Un ejemplo de alfabeto es el alfabeto latino $\Sigma = \{a, b, c, d, ..., z\}$, cuyos símbolos son los caracteres latinos.

Cadenas

Cadena

Una cadena, que denotaremos como w, es un arreglo finito de símbolos:

$$w = (a_1, a_2, ..., a_k) := a_1 a_2 \cdots a_k$$

tal que para toda i, $a_i \in \Sigma$.

Por ejemplo, la cadena `gato` se conforma de los símbolos g,a,t,o del alfabeto latino.

Cadena vacía

A la cadena que no contiene ningún símbolo se le denota como ϵ. Podemos pensarlo como un arreglo vacío $\epsilon := ()$.

La cadena vacía será de esencial importancia.

Operación de concatenación

Concatenación

La operación de concatenación entre dos cadenas $x = a_1 a_2 \cdots a_n$ e $y = b_1 b_2 \cdots b_m$, es la operación $\cdot$ tal que:

$$x \cdot y = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m$$

Propiedades de la concatenación

La operación de concatenación entre cadenas $\cdot$ cumple que para cualquieras cadenas w, x, y:

1. $\epsilon \cdot w = w \cdot \epsilon = w$ (Neutro)
2. $w \cdot (x \cdot y) = (w \cdot x) \cdot y$ (Asociatividad)

Otra definición de cadena

Cadena

Se puede definir una cadena w sobre un alfabeto Σ de manera recursiva de la siguiente forma:

1. ϵ y $a \in \Sigma$ son cadenas.
2. Si w es una cadena, entonces $w \cdot a$ para algún $a \in \Sigma$ es cadena.

Por ejemplo, sabemos que g es una cadena, pues es un símbolo del alfabeto latino. También ga es una cadena, pues $ga = g \cdot a$, con a parte del alfabeto latino.
Si gato es una cadena y s un símbolo del alfabeto, gato·s=gatos es una cadena.

Longitud de cadenas

Longitud de cadena

La longitud de una cadena $w = a_1 a_2 \cdots a_k$ es el número de símbolos que contiene la cadena. Podemos definir la longitud de la cadena,$|w|$, como:

1. $|\epsilon| = 0$
2. Si w es una cadena y $a \in \Sigma$, entonces $|w \cdot a| = |w| + 1$

Por ejemplo, la longitud de la cadena `gato` puede calcularse como:

- Ya que $g = \epsilon \cdot g$ tenemos que $|g| = |\epsilon \cdot g| = 0 + 1 = 1$.
- $|ga| = |g \cdot a| = |g| + 1 = 1 + 1 = 2$
- $|gat| = |ga \cdot t| = |ga| + 1 = 2 + 1 = 3$
- $|gato| = |gat \cdot o| = |gat| + 1 = 3 + 1 = 4$

Longitud de cadena

Proposición

La función para obtener la longitud de la cadena cuenta, en efecto, el número de símbolos de una cadena.

Para demostrarlo, haremos inducción sobre la cadena.
Base: Si $w = \epsilon$, entonces la cadena no tiene ningún símbolo de Σ, por lo que $|\epsilon| = 0$ es correcto. Si $w = a \in \Sigma$, entonces la cadena sólo contiene un símbolo, por tanto $|a| = |\epsilon \cdot a| = |\epsilon| + 1 = 1$ es correcto.
Inducción: Supóngase que la cadena $w = x \cdot a$, con $x \in \Sigma^*$ y $a \in \Sigma$. Entonces, $|w| = |x| + 1$, por hipótesis de inducción, la función calcula el número de símbolos de x, sea $|x| = n$, por tanto, $|w| = n + 1$ considera el símbolo que se ha concatenado.

Longitud en concatenación

Proposición

Si x e y son cadenas, entonces:

$$|x \cdot y| = |x| + |y|$$

Prueba.

Por inducción sobre la longitud de y.

- Si $|y| = 0$, necesariamente $y = \epsilon$. Por lo que: $|x \cdot y| = |x \cdot \epsilon| = |x| + 0 = |x|$, pues $x \cdot \epsilon = x$.
- Supóngase que la proposición es cierta para cadenas de longitud n. Para $|y| = n + 1$, sea $y = w \cdot a$, tal que $a \in \Sigma$ y $|w| = n$. Entonces:

$$\begin{aligned} |x \cdot y| &= |x \cdot (w \cdot a)| = |(x \cdot w) \cdot a| = |(x \cdot w)| + 1 = (|x| + |w|) + 1 \\ &= |x| + (|w| + 1) = |x| + (n + 1) = |x| + |y| \end{aligned}$$

□

Cadenas y listas

Las cadenas se pueden estructurar como listas, esto es:

$$w = [a_1, a_2, ..., a_n]$$

De tal forma que $\epsilon = []$ y $x \cdot a = x + [a]$. También se pueden aplicar operaciones similares a las listas, como acceder a elementos:

$$w[k] = a_k$$

Donde a_k es el k-ésimo símbolo de la cadena.

Conjuntos de cadenas

Concatenación de conjuntos

Dado conjuntos (alfabetos) Σ_1 y Σ_2, definimos la concatenación como:

$$\Sigma_1 \cdot \Sigma_2 = \{x \cdot y : x \in \Sigma_1, y \in \Sigma_2\}$$

Potencias del alfabeto

Los conjuntos de cadenas también pueden operarse entre sí. Las potencias del alfabeto Σ se definen como:

1. $\Sigma^0 = \{\epsilon\}$
2. $\Sigma^{n+1} = \Sigma^n \cdot \Sigma$

Definiremos los conjuntos de cadenas de longitud n como sigue:

$$\Sigma^n := \{w : |w| = n\}$$

Σ^*

Σ^*

El conjunto de todas las cadenas posibles sobre un alfabeto Σ, se denota como Σ^* y puede definirse como:

$$\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k$$

Este conjunto contiene al elemento ϵ. Pero podemos definir al conjunto de las cadenas sin la cadena vacía:

$$\begin{aligned} \Sigma^+ &= \bigcup_{k=1}^{\infty} \Sigma^k \\ &= \Sigma^* \setminus \{\epsilon\} \end{aligned}$$

Σ^*

Teorema

El conjunto de todas las cadenas Σ^* es numerable. Esto es $|\Sigma^*| = \aleph_0$

Prueba.

Sea $h : \Sigma \to \mathbb{N}$ la función que asocia a cada símbolo un natural $(h(a_i) = i \in \mathbb{N}, 1 \leq i \leq n)$. Defínase la función $f : \Sigma^* \to \mathbb{N}$ de la siguiente forma:

$$f(\epsilon) = 0$$
$$f(w \cdot a) = n \cdot f(w) + h(a)$$

La función f es biyectiva (¿por qué?), de lo que concluimos que Σ^* es numerable. □

Lenguajes

Lenguaje

Un lenguaje L es un subconjunto de cadenas sobre un alfabeto Σ. Esto es $L \subseteq \Sigma^*$.

La clase de todos los lenguajes sobre un alfabeto Σ, es el conjunto de todos los subconjunto de Σ^*; i.e., $\mathcal{P}(\Sigma^*)$.

Ejemplo: Si $\Sigma = \{0, 1\}$ es un alfabeto. Son lenguajes:

- $L = \{\epsilon\}$, pues $\epsilon \in \Sigma^*$.
- $L = \emptyset$, pues $\emptyset \subset \Sigma^*$
- $L = \{0, 1\}$, pues $\Sigma \subset \Sigma^*$.
- $L = \Sigma^*$, pues $\Sigma^* \subseteq \Sigma^*$.
- $L = \{w \in \Sigma^* : dec(w) \text{ es primo}\} = \{10, 11, 101, 111, 1011, ...\}$

Subcadenas

Subcadena

Dada una cadena $w \in \Sigma^*$, una subcadena de w es una cadena x tal que $\exists y, z \in \Sigma^*$ tal que $w = y \cdot x \cdot z$.

Afijos

Un afijo de una cadena w es una subcadnea:

- $x \in \Sigma^*$ es un **prefijo** de w si $\exists y \in \Sigma^*$ tal que $w = x \cdot y$.
- $y \in \Sigma^*$ es un **sufijo** de w si $\exists x \in \Sigma^*$ tal que $w = x \cdot y$.

Lenguaje de palíndromos

Operación de inverso

Dado una cadena $w \in \Sigma^*$, el inverso de w, denotado como w^R se define recursivamente como:

- $\epsilon^R = \epsilon$
- $(w \cdot a)^R = a \cdot w^R$

Palíondromo

Un palíndromo es una cadena w tal que cumple que

$$w = w^R$$

El conjunto de todos los palíndromos sobre un alfabeto,

$$L = \{w \in \Sigma^* : w = w^R\}$$

es un lenguaje.

Operaciones sobre lenguajes

Operaciones sobre lenguajes

Si $L_1, L_2 \subseteq \Sigma^*$ son dos lenguajes, entonces también son lenguajes:

1. $L_1 \cup L_2 = \{w \in \Sigma^* : w \in L_1 \vee w \in L_2\}$ — Unión
2. $L_1 \cap L_2 = \{w \in \Sigma^* : w \in L_1 \wedge w \in L_2\}$ — Intersección
3. $L_1^c = \{w \in \Sigma^* : w \notin L_1\}$ — Complemento
4. $L_1 \cdot L_2 = \{w_1 \cdot w_2 \in \Sigma^* : w_1 \in L_1, w_2 \in L_2\}$ — Concatenación
5. $L_1^0 = \{\epsilon\}$ — Potencia (base)
6. $L_1^{n+1} = L_1^n \cdot L_1$ — Potencia
7. $L_1^* = \bigcup_{n=0}^{\infty} L_1^n$ — Cerradura Kleene
8. $L^R = \{w^R : w \in L_1\}$ — Reverso

Propiedades de los lenguajes

Dado el conjunto de todos los lenguajes con la operación de concatenación, $(\mathcal{P}(\Sigma^*), \cdot)$, se cumplen las siguientes propiedades:

1. Sea $L_\epsilon = \{\epsilon\}$, entonces para todo L lenguaje, $L \cdot L_\epsilon = L_\epsilon \cdot L = L$. — Identidad
2. Para todo L lenguaje, $L \cdot \emptyset = \emptyset$ — Dominante
3. Para todos L_1, L_2, L_3 lenguajes, $L_1 \cdot (L_2 \cdot L_3) = (L_1 \cdot L_2) \cdot L_3$ — Asociatividad
4. Para todos L_1, L_2, L_3 lenguajes, $L_1 \cdot (L_2 \cup L_3) = (L_1 \cdot L_2) \cup (L_1 \cdot L_3)$ — Distributividad

Reglas de una gramática

Algunos lenguajes pueden ser infinitos, por lo que no es conveniente representarlos enumerando sus elementos. Existen diferentes técnicas de representar los lenguajes. Una de ellas son las **gramáticas**.
De manera informal, una **gramática** es una serie de reglas que nos dicen cómo constuir las cadenas de un lenguaje.

Sistema semi-Thue

Un sistema semi-Thue es una tupla (Σ, R) tal que Σ es un alfabeto y $R \subseteq \Sigma^+ \times \Sigma^*$ una relación. Denotaremos a los elementos en R como $x \to y$, con $x \in \Sigma^+$ e $y \in \Sigma^*$

Gramática

Gramática

Una gramática es una 4-tupla $G = (\Sigma, \Delta, S, R)$ donde:

- Σ y Δ son alfabetos, el terminal y no terminal, respectivamente.
- $S \in \Delta$ es el símbolo inicial.
- $(\Delta \cup \Sigma, R)$ es un sistema semi-Thue.

Derivación

Dada una gramática $G = (\Sigma, \Delta, S, R)$ y dos cadenas x, y, decimos en la gramática x deriva en y, denotado $x \Longrightarrow_G y$, sii $x = \alpha X \beta$, $y = \alpha Y \beta$, con $\alpha, \beta, Y \in (\Delta \cup \Sigma)^*$, $X \in (\Delta \cup \Sigma)^+$ y existe $X \to Y \in R$.

Lenguaje de la gramática

Cerradura transitiva-reflexiva de derivación

Dada una gramática $G = (\Sigma, \Delta, S, R)$ con derivaciones $\Longrightarrow_G$, la cerradura transitiva-reflexiva de esta, denotada $\Longrightarrow_G^*$, cumple:

- Para toda cadena w, $w \Longrightarrow_G^* w$.
- Si $w \Longrightarrow_G x$ y $x \Longrightarrow_G y$, entonces $w \Longrightarrow_G^* y$.

Aceptación de la gramática

Dada una gramática $G = (\Sigma, \Delta, S, R)$, decimos que una cadena $w \in \Sigma^*$ es aceptada por la gramática si $S \Longrightarrow_G^* w$. Si w es aceptada por la gramática G, lo denotamos como $G \models w$.

Lenguaje generado por una gramática

Si $G = (\Sigma, \Delta, S, R)$ es una gramática, el lenguaje generado por la gramática es:

$$L(G) = \{w \in \Sigma^* : G \models w\}$$

Ejemplo

Definamos una gramática $G = (\Sigma, \Delta, S, R)$, donde $\Sigma = \{0, 1\}$, $\Delta = \{S\}$, S es el inicial, y las reglas R están dadas por:

$$S \to 0S0|1S1$$
$$S \to \epsilon|0|1$$

La cadena 10101 es derivada de la gramática como:

$$S \Longrightarrow_G 1S1 \Longrightarrow_G 10S01 \Longrightarrow_G 10101$$

Por tanto $S \Longrightarrow^*_G 10101$ y concluimos que $G \models 10101$.
Más aún, se puede comprobar que esta gramática describe el lenguaje de los palíndromos.

Ejemplo

Proposición

La gramática G anterior define el lenguaje de palíndromos $L = \{w : w = w^R\}$

Prueba.

Por inducción (fuerte) sobre la longitud de la cadena w.

1. Si $|w| = 0$, entonces $w = \epsilon$ palíndromo, y se aplica la derivación $S \Longrightarrow \epsilon$, si $|w| = 1$, entonces $w = 0$ ó $w = 1$ y se usa $S \Longrightarrow 0$ ó $S \Longrightarrow 1$.
2. Supóngase cierta la afirmación para cadenas de longitud n. Supóngase $w = w^R$ con $|w| = n + 1$, entonces $w = 0x0$ ó $w = 1x1$, con $x \in \Sigma^*$. Ya que $|x| < n + 1$ y x debe ser palíndromo, por hipótesis de inducción, $S \Longrightarrow_G^* x$, pero tenemos que $S \Longrightarrow_G 0S0 \Longrightarrow_G^* 0x0$, por lo que $S \Longrightarrow_G^* 0x0$. De forma similar vemos que $S \Longrightarrow_G^* 1x1$, por lo que $G \models w$.

Concluimos que la gramática puede derivar cualquier palíndromo. Por lo que $L \subseteq L(G)$. □

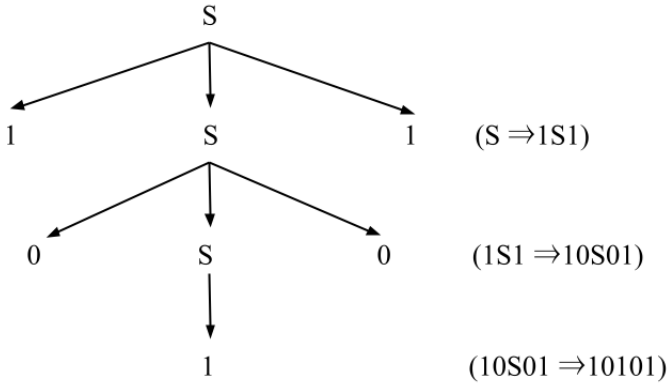
Árboles de derivación

Un árbol de derivación es un árbol $T = (V, E)$ que representa la derivación de una cadena a partir de una gramática, tal que:

1. Toda $n \in V$ nodo está asociado a un símbolo en $\Sigma \cup \Delta \cup \{\epsilon\}$.
2. La raíz del árbol está asociado al símbolo inicial S.
3. Si $X \to Y_1 Y_2 \cdots Y_n$ es una producción en R, entonces el nodo asociado a X tiene como hijos a $Y_1, Y_2, ..., Y_n$.
4. Los símbolos terminales Σ, o ϵ, no tienen hijos. Son hojas del árbol.

Ejemplo de árbol de derivación

La gramática de los palíndromos puede generar la cadena 10101 con las derivaciones propuestas. Esto da pie al árbol de derivación:



Modelo de procesamiento de cadenas

Los **autómatas** son modelos de máquinas abstractas. Al estudiar estos modelos nos interesa:

1. Estudiar sus **capacidades**, qué clases de objetos pueden manejar.
2. Estudiar formas de **sintetizarlas**, cómo acoplar distintos modelos de éstas.
3. Diseñar **compiladores** como la interconexión de varios modelos.
4. **Simular** estos modelos con programas de computadora.

En principio, hablaremos de tres tipos de modelos:

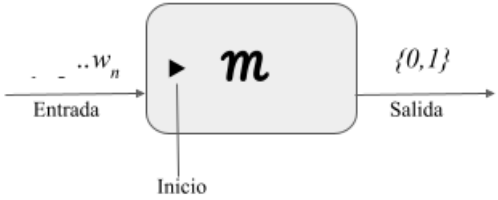
1. Aceptadores
2. Generadores
3. Transductores

Aceptadores

Aceptador

Un aceptador A es una máquina (autómata) diseñada para aceptar o rechazar cadenas de símbolos, $w \in \Sigma^*$. Formalmente, $A = (Q, \Sigma, q_0, F, \delta)$, tal que Q son estados, Σ alfabeto, $q_0 \in Q$ estado inicial, y δ función de transición.

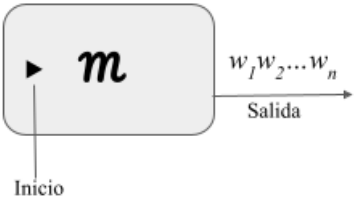
Un aceptador emite un 1 si la cadena de entrada llega a un final, o un 0 si no.



Generadores

Generador

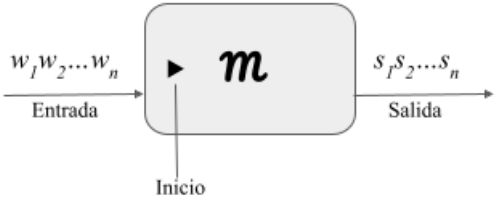
Un generador G es una máquina diseñada para generar cadenas a partir de un estado inicial; generalmente, es un modelo no-determinista, pues puede generar diferentes cadenas cada vez que trabaja. Formalmente, $G = (Q, \Sigma, q_0, \lambda)$, donde $\lambda : Q \to Q \times \Sigma^*$ es una función no determinista.



Transductores

Transductor

Un transductor toma una cadena de entrada en un lenguaje y emite una cadena de salida en otro lenguaje.
Formalmente, $T = (Q, \Sigma, \Delta, q_0, \sigma)$, donde Q es un conjunto de estados, $q_0 \in Q$ el estado inicial, Σ y Δ el alfabeto de entrada y salida, y $\sigma : Q \times \Sigma \times \Delta \to Q$.



Relación de autómatas y lenguajes

Se estudiarán diferentes tipos de lenguajes y de autómatas. Ahora podemos ver los lenguajes otenidos y su relación con aceptadores y reconocedores:

Lenguaje	**Autómata**	**Memoria**	**Regla**
General	Máquinas de Turing	Cinta infinita	$\alpha \to \gamma$
Dependiente del contexto	Linealmente acotados	Cinta acotada	$\alpha X \beta \to \alpha \gamma \beta$
Libre de contexto	Con pila	Pila	$X \to \gamma$
Regular	Finitos	Estado	$X \to aX \vert a$