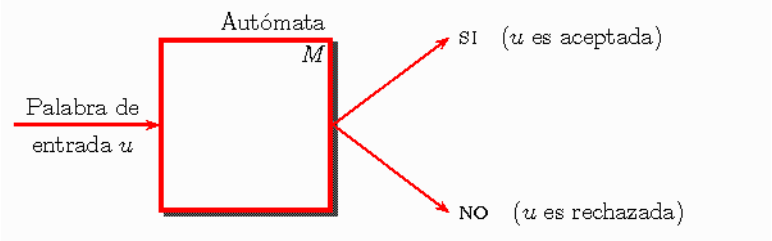
Autómatas y Lenguajes Formales 2019-II
Facultad de Ciencias UNAM*
Nota de Clase 3: Autómatas finitos deterministas

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

12 de marzo de 2019

Como hemos comentado en clase, un autómata es una representación abstracta de una máquina. Sobre todo, un autómata finito es una máquina abstracta que procesa cadenas aceptándolas o rechazándolas.



Ahora estudiaremos los autómatas más sencillos que están en correspondencia con los lenguajes regulares ya expuestos.

1. Autómatas Finitos Deterministas

Definición 1 *Un autómata finito determinista (AFD) es una quintupla* $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ *donde*

- Q *es un conjunto finito de estados.*
- Σ *es el alfabeto de entrada.*
- $\delta : Q \times \Sigma \to Q$ *es la función de transición.*
- $q_0 \in Q$ *es el estado inicial.*
- $F \subseteq Q$ *es el conjunto de estados finales.*

Los autómatas finitos definidos se denominan **autómatas finitos deterministas** ya que para cada estado q y para cada símbolo $a \in \Sigma$, la función de transición $\delta(q, a)$ *siempre* está definida. Es decir, la función de transición δ *determina unívocamente* la acción que el autómata realiza cuando se encuentra en un estado q procesando un símbolo a.

*Material elaborado en el marco del proyecto PAPIME PE102117

Definición 2 *Dado un autómata M, el lenguaje aceptado o reconocido por M se denota $L(M)$ y se define por*

$$L(M) := \{w \in \Sigma^* \mid M \text{ se detiene al procesar } w \text{ en un estado } q \in F\}$$

Ejemplo: Sean $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2\}$ donde q_0 es el estado inicial y el conjunto $F = \{q_0, q_2\}$ son los estados de aceptación. La función de transición δ está definida por alguna de las siguientes:

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_1

$$\delta(q_0, a) = q_0 \qquad \delta(q_0, b) = q_1$$
$$\delta(q_1, a) = q_1 \qquad \delta(q_1, b) = q_2$$
$$\delta(q_2, a) = q_1 \qquad \delta(q_2, b) = q_1$$

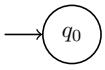
Del ejemplo anterior se pueden ver dos formas de presentar la función de transición: tabla de estados con símbolos o la definición estándar usada para funciones. También es posible dar una presentación visual del autómata mediante diagramas para representar estados y las transiciones entre ellos.

1.1. Diagrama de estados

Un autómata finito se puede representar por medio de un grafo dirigido y etiquetado. Recuérdese que un **grafo** es un conjunto de vértices o nodos unidos por aristas o conectores; si las aristas tienen tanto dirección como etiquetas, el grafo se denomina **grafo dirigido y etiquetado** o **digrafo etiquetado**.
Para el caso de los autómatas finitos, esta representación está modificada de la siguiente forma:

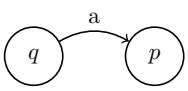
- Los vértices o nodos son los estados del autómata.
- Un estado q se representa por:

- El estado inicial q_0 se representa por:

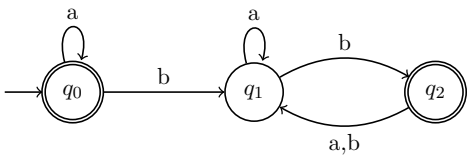


- Un estado final q_f se representa por:

- La transición $\delta(q, a) = p$ se representa en la forma:



Ejemplo: Veamos la representación gráfica del autómata del ejemplo anterior:



Para autómatas deterministas se adopta una y sólo una de las siguientes convenciones con respecto a los diagramas de estados:

- Como se dijo antes, la función de transición es **total**: cada estado tiene una transición por cada símbolo. Así se puede considerar la existencia de un **estado de error** para las cadenas que serán rechazadas
- Alternativamente se puede asumir que las aristas no dibujadas explícitamente conducen al estado de no-aceptación llamado **estado de error**.
  Es decir, en el diagrama de estados se indican únicamente las aristas que conduzcan a trayectorias de aceptación. Esto permite simplificar considerablemente los diagramas.

1.2. Autómatas y lenguajes

Los autómatas son máquinas que nos permiten reconocer lenguajes y la forma de hacerlo es através de un procesamiento de cadenas que permite "recordar" algunas características de dichas cadenas. El diseño de estas máquinas es un problema importante a abordar:

Problema: Dado un lenguaje $L \subseteq \Sigma^\star$ diseñar un AFD M
que acepte exactamente a L, es decir, tal que $L(M) = L$.

Una estrategia de ensayo y error es inconveniente ya que las posibilidades son demasiadas al considerar los símbolos del alfabeto dado y las características del lenguaje a detallar. Además se deben respetar las siguientes características para resolver el problema anterior:

- El autómata debe ser **completo**: debe aceptar todas las palabras de L, es decir que es necesario que $L \subseteq L(M)$.
- El autómata debe ser **correcto**: debe aceptar sólo palabras de L, es decir que es necesario que $L(M) \subseteq L$.

Esta doble contención asegurará que ambos lenguajes sean iguales y que el autómata acepte exactamente las cadenas del lenguaje.

Hay muchos problemas de interés sobre autómatas finitos, nos interesa resolver en este momento los siguientes:

- Dado un autómata M determinar formalmente el lenguaje aceptado por M.
- Dado un lenguaje regular L diseñar un autómata finito determinista M que acepte o reconozca a L, es decir, tal que $L(M) = L$.

Más adelante se demostrará que estos problemas *siempre* tienen solución. Por ejemplo, el autómata discutido anteriormente, reconoce al lenguaje $a^* + a^*ba^*ba^*$ es decir, el lenguaje regular $L = \{w \in \{a,b\}^\star \mid w$ tiene u número par de bes $\}$

El primer paso para resolver los problemas anteriores es la formalización de la noción de procesamiento, *grosso modo* se debe abstraer la idea de que las cadenas serán "leidas" por el autómata y harán que éste se detenga en un estado. Si en el estado es final, las cadenas serán aceptadas y serán rechazadas en otro caso. Con esto se puede determinar cuál es el lenguaje que acepta el autómata.

2. Lenguaje de Aceptación

La idea de diseñar una máquina abstracta para el procesamiento de cadenas, está captado por el funcionamiento de un autómata. De manera informal un autómata M **reconoce** o acepta una cadena w si:

1. Se consumen todos los símbolos de w al comenzar en el estado inicial, generalmente el llamado q_0, siguiendo las transiciones de acuerdo a la función δ.
2. Al terminar, el estado actual de la máquina es final.

El lenguaje aceptado es entonces el mismo que se describió al principio de esta nota mediante una notación por comprensión. Esta definición un tanto informal se puede hacer estricta mediante una extensión de la función de transición para denotar el procesamiento de una cadena en su totalidad.

Definición 3 *Sea $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ un AFD. La función de transición δ se extiende a cadenas mediante una función*

$$\delta^\star : Q \times \Sigma^\star \to Q$$

llamada la función de transición extendida de M y es definida recursivamente como sigue:

- $\delta^\star(q, \varepsilon) = q$
- $\delta^\star(q, wa) = \delta\big(\delta^\star(q, w), a\big)$*, o alternativamente* $\delta^\star(q, aw) = \delta^\star\big(\delta(q, a), w\big)$.

De la definición se sigue que $\delta^\star(q, w)$ devuelve el estado en el que la máquina **termina** al procesar w. Dado que $\delta^\star$ es una extensión de δ es usual *sobrecargar* la operación y escribir δ en lugar de $\delta^\star$.

La definición informal de lenguaje de aceptación de un autómata puede ahora formalizarse haciendo referencia a la función de transición extendida:

Definición 4 *El lenguaje de aceptación se define como:*

$$L(M) = \{w \in \Sigma^\star \mid \delta^\star(q_0, w) \in F\}$$

De esta manera es posible verificar formalmente la completud y correctud del diseño de un autómata M para un lenguaje L:

- Correctud ($L(M) \subseteq L$): Si $\delta^\star(q_0, w) \in F$ entonces $w \in L$.
- Completud ($L \subseteq L(M)$): Si $w \in L$ entonces $\delta^\star(q_0, w) \in F$.

Debemos recordar que estas dos propiedades son indispensables para el diseño de una máquina que reconozca un lenguaje.

3. Diseño de autómatas finitos deterministas

La noción de un autómata como máquina abstracta con memoria nos proporciona una serie de estrategias a seguir para el diseño de autómatas:

- Se debe determinar explícitamente qué condición "recuerda" cada estado.
- La única memoria que tiene un AFD son los estados.
- Primero se propone un conjunto de estados que "recuerden" condiciones importantes.
- Después se proponen las transiciones para cambiar de un estado a otro.
- Finalmente se determina cuáles estados son finales, observando qué condiciones de estado corresponden con la definición del lenguaje aceptado.

A continuación se detallan algunas estrategias bien establecidas para obtener autómatas.

3.1. Diseño por conjuntos de estados

Muchas veces es preferible generalizar condiciones para definir grupos de estados en lugar de dar condiciones estado por estado. De esta forma se disminuye la posibilidad de error y se facilita la solución de un problema complejo.
Las condiciones para grupos de estados deben ser:

- Excluyentes: un grupo de estados debe cumplir únicamente una condición.
- Comprensivas: los grupos cumplen todos los casos posibles.

Ejemplo: Diseñar un AFD para

$$L = \{w \in \{0,1\}^\star \mid w \text{ no contiene 11 pero sí 00}\}$$

Un análisis informal nos permite distinguir los grupos de estados según las siguientes condiciones y subcasos:

1. La palabra a ser consumida no contiene ni 00 ni 11:
   - no se han leído símbolos
   - se leyó un 0
   - se leyó un 1
2. La palabra contiene 00 pero no 11:
   - se leyó otro 0
   - se leyó un 1
3. La palabra contiene la subcadena 11.
   - Se leyó otro 1.

   Este análisis da un conjunto de seis estados, tres en el primer grupo, dos en el segundo y uno en el tercero.

3.2. Diseño por complemento

Dado un lenguaje L a veces es más facil diseñar un autómata para el complemento $\overline{L} = \Sigma^\star - L$:

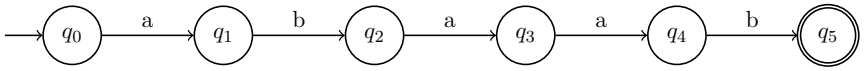
> Si $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ reconoce a L, es decir $L(M) = L$ entonces $M^c = \langle Q, \Sigma, \delta, q_0, Q - F\rangle$ reconoce a $\Sigma^\star - L$, es decir $L(M^c) = \overline{L}$.

Este método sólo funciona para autómatas deterministas.

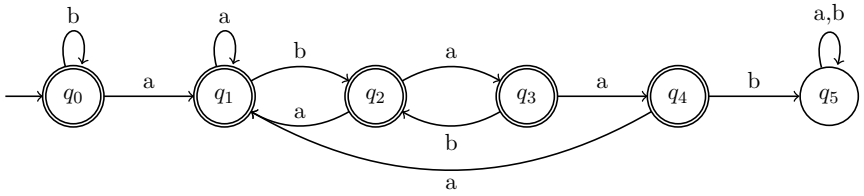
Ejemplo: Diseñar un AFD para

$$L = \{w \in \{a, b\}^\star \mid w \text{ no contiene } abaab\}$$

1. Primero se diseña la parte de M^c que acepta la cadena $abaab$:



2. Se completa el autómata anterior y se obtienen los complementos de los estados para obtener M:



3.3. Modularización

Algunas veces la especificación de un lenguaje L tiene una forma lógica que permite descomponerlo en lenguajes más sencillos. De esta manera el diseño de un AFD se puede modularizar.

- Por ejemplo si $L = L_1 \cup L_2$ basta construir autómatas para L_1 y L_2 y ejecutarlos en paralelo.
- Lo mismo sucede si $L = L_1 \cap L_2$ o si $L = L_1 - L_2$.

Veamos la definición formal para obtener los anteriores:

Definición 5 *Suponer que $M_1 = \langle Q_1, \Sigma, \delta_1, q_0, F_1\rangle$ y $M_2 = \langle Q_2, \Sigma, \delta_2, p_0, F_2\rangle$ son dos autómatas que aceptan los lenguajes L_1 y L_2 respectivamente. Sea $M = \langle Q, \Sigma, \delta, r_0, F\rangle$ un autómata donde $Q = Q_1 \times Q_2$, el estado inicial es $r_0 = (q_0, p_0)$, la función de transición se define como*

$$\delta((q, p), a) = (\delta_1(q, a), \delta_2(p, a))$$

y los estados finales como:

1. *Si $F = \{(q, p) \mid q \in F_1 \text{ o } p \in F_2\}$ entonces M acepta el lenguaje $L_1 \cup L_2$.*
2. *Si $F = \{(q, p) \mid q \in F_1 \text{ y } p \in F_2\}$ entonces M acepta el lenguaje $L_1 \cap L_2$.*
3. *Si $F = \{(q, p) \mid q \in F_1 \text{ y } p \notin F_2\}$ entonces M acepta el lenguaje $L_1 - L_2$.*

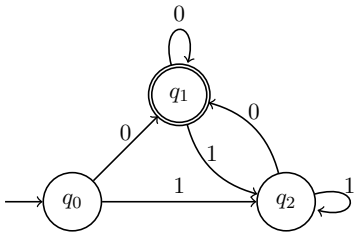
Una estrategia a seguir es calcular la tabla de transiciones para el nuevo autómata y renombrar los estados nuevos. Así mismo, si existen estados que no están conectados se pueden eliminar.

Ejemplo: Sea $L = \{w \in \{0,1\}^\star \mid w \text{ acaba en 0 y 11 no es subcadena de } w\}$. Para obtener un autómata que reconozca el lenguaje se descompone como $L = L_1 - L_2$ donde

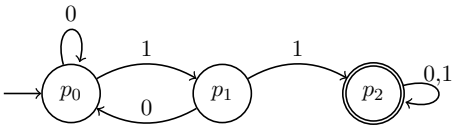
$$L_1 = \{w \in \{0,1\}^\star \mid w \text{ acaba en } 0\} \qquad L_2 = \{w \in \{0,1\}^\star \mid 11 \text{ es subcadena de } w\}$$

A continuación se construyen los autómatas para reconocer los respectivos lenguajes y después se obtiene M usando la definición anterior, se proporciona la tabla de transiciones.

M_1 :



M_2 :



M :

$\delta_1 \times \delta_2$	0	1	renombre	
(q_0, p_0)	(q_1, p_0)	(q_2, p_1)	r_0	edo. inicial
(q_0, p_1)	(q_1, p_0)	(q_2, p_2)	r_1	edo. no conectado
(q_0, p_2)	(q_1, p_2)	(q_2, p_2)	r_2	edo. no conectado
(q_1, p_0)	(q_1, p_0)	(q_2, p_1)	r_3	edo. final
(q_1, p_1)	(q_1, p_0)	(q_2, p_2)	r_4	edo. final pero no conectado
(q_1, p_2)	(q_1, p_2)	(q_2, p_2)	r_5	
(q_2, p_0)	(q_1, p_0)	(q_2, p_1)	r_6	edo. no conectado
(q_2, p_1)	(q_1, p_0)	(q_2, p_2)	r_7	
(q_2, p_2)	(q_1, p_2)	(q_2, p_2)	r_8	