Autómatas y Lenguajes Formales 2019-I
Facultad de Ciencias UNAM
Nota de Clase 5

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

28 de agosto de 2018

Como hemos visto, la relación entre autómatas finitos y lenguajes regulares parece de equivalencia, las nociones detrás de ambos conceptos la suguieren. En esta nota demostraremos que ese es el caso utilizando el método propuesto por Kleene [1]

1. Teorema de Kleene

Teorema 1 *Un lenguaje es regular si y sólo si es aceptado por un autómata finito.*

Demostración. La prueba es en dos partes:

I Síntesis: Dado un lenguaje regular L, existe un autómata finito M tal que $L = L(M)$.

II Análisis: Dado un autómata finito M, existe una expresión regular α tal que $L(M) = L(\alpha)$. Es decir, $L(M)$ es regular.

A continuación se abordarán dos teoremas que demuestran al anterior. Para ello recordemos que las expresiones regulares están en correspondencia con los lenguajes regulares y de esta forma podremos empatar también a las expresiones regulares con los autómatas finitos.

1.1. Teorema de Síntesis de Kleene

En esta sección se demostrará una parte de la doble implicación del teorema de Kleene: se pasará de expresiones regulares a autómatas finitos.

Este teorema se denomina de síntesis ya que se proporcionará una máquina para reconocer un lenguaje regular dado. Es decir, se sintetizará un autómata finito analizando la forma de una expresión regular que genera al lenguaje dado.

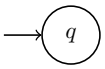
Teorema 2 *Dada una expresión regular α existe un autómata finito M tal que $L(\alpha) = L(M)$.*

Demostración. La demostración es *constructiva* y se hará mediante inducción sobre las expresiones regulares, es decir proporcionando un autómata que *reconzca* cada caso de una expresión regular.

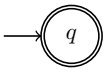
[1] Stephen Cole Kleene fue un matemático estadounidense, alumno de Alonzo Church. También es conocido por iniciar la teoría de la recursión que fue usada para los fundamentos de la Teoría de la Computación como la noción de computabilidad.

Base de la Inducción

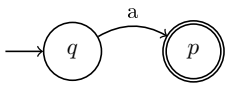
El caso en que $\alpha = \varnothing$, el siguiente autómata reconoce a $L(\alpha)$:



Caso en que $\alpha = \boldsymbol{\varepsilon}$. Entonces el siguiente autómata reconoce a $L(\alpha)$:



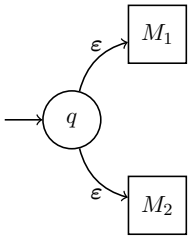
Para $\alpha = a$, con $a \in \Sigma$ se tiene el siguiente autómata que reconoce a $L(\alpha)$:



Hipótesis de inducción

Sean M_1, M_2 dos autómatas que reconocen los lenguajes $L(\alpha_1)$ y $L(\alpha_2)$ respectivamente.

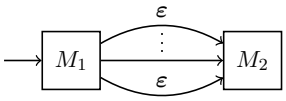
Paso Inductivo



Caso en que $\alpha = \alpha_1 + \alpha_2$. El siguiente autómata reconoce a $L(\alpha)$:

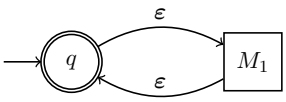
donde M_1, M_2 son autómatas dados por la hipótesis de inducción y las transiciones $\boldsymbol{\varepsilon}$ van hacia los estados iniciales de cada uno de ellos.

Para el caso en que $\alpha = \alpha_1\alpha_2$ entonces el siguiente autómata reconoce a $L(\alpha)$:



donde M_1, M_2 son autómatas que reconocen a $L(\alpha_1), L(\alpha_2)$ dados por la hipótesis de inducción, el estado inicial es el de M_1 y las transiciones $\boldsymbol{\varepsilon}$ van de los estados finales de M_1 hacia el inicial en M_2.

Finalmente el caso $\alpha = \alpha_1^\star$ tiene al siguiente autómata que reconoce a $L(\alpha)$:



donde M_1 es un autómata que reconoce a $L(\alpha_1)$ dado por la hipótesis de inducción y las transiciones $\boldsymbol{\varepsilon}$ conectan el estado inicial y los finales con el nuevo estado q.

1.2. Teorema de Análisis de Kleene

Ahora demostraremos la segunda parte en donde un autómata finito implica una expresión regular, la noción detrás de esta parte es que analizaremos los lenguajes acumulados de cada estado para generar una expresión regular.

Teorema 3 *Dado un autómata finito M existe una expresión regular α tal que $L(M) = L(\alpha)$. Es decir, $L(M)$ es regular.*

Demostración. Existen diversas demostraciones, nosotros usaremos el método de ecuaciones características usando el Lema de Arden.

1.2.1. Lema de Arden

Este lema extrae un conjunto de ecuaciones para determinar el lenguaje de aceptación de una máquina. Primero veamos la definicón de dichas ecuaciones y después el método para obtener las ecuaciones dado un autómata.

Definición 1 *Sean $A, B \subseteq \Sigma^\star$ y X una variable:*

- *Una ecuación lineal derecha para X es una expresión de la forma:*

$$X = AX + B$$

- *Análogamente, una ecuación lineal izquierda es una expresión de la forma:*

$$X = XA + B$$

Donde el símbolo $+$ denota a la unión de lenguajes.

Lema 1*: Lema de Arden* *Sean $A, B \subseteq \Sigma^\star$ dos lenguajes y $X = AX + B$ una ecuación lineal derecha. Entonces*

1. *$A^\star B$ es una solución de la ecuación, es decir, $A^\star B = A(A^\star B) + B$.*
2. *Si C es otra solución entonces $A^\star B \subseteq C$, es decir, $A^\star B$ es la solución mínima.*
3. *Si $\varepsilon \notin A$ entonces $A^\star B$ es la única solución.*

Esta parte mostrará que un autómata finito implica una expresión regular, esto se hará por medio de un sistema de ecuaciones a partir de un AFN, para ello se abstraerá la noción del lenguaje aceptado desde un estado particular del autómata y no necesariamente del inicial.

Definición 2 *Dado un AFN $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ tal que $Q = \{q_0, \ldots, q_n\}$. Definimos los siguientes conjuntos:*

- *El conjunto de cadenas que se aceptan desde el estado q_i, para cualquier $1 \leq i \leq n$:*

$$L_i = \{w \in \Sigma^\star \mid \delta^\star(q_i, w) \cap F \neq \varnothing\}$$

- *L_0 es el lenguaje aceptado por M, es decir, $L_0 = L(M)$.*

En general no es sencillo calcular directamente los conjuntos L_i. Para obtener una expresión regular completa respecto al autómata, se obtendrá un sistema de ecuaciones a partir de un AFN. Al resolverlo, L_0 será el lenguaje que reconoce el autómata como se mencionó arriba.
El sistema de ecuaciones se define usando:

1. El conjunto de símbolos de Σ tal que existe una transición del estado q_i al estado q_j, para cualesquiera $1 \leq i, j \leq n$, con n el total de estados:

$$X_{i,j} = \{a \in \Sigma \mid q_j \in \delta(q_i, a)\}$$

2. El conjunto auxiliar Y_i que indica si $\boldsymbol{\varepsilon}$ es aceptada desde q_i

$$Y_i = \begin{cases} \{\boldsymbol{\varepsilon}\} & \text{si } q_i \in F \\ \varnothing & \text{en otro caso} \end{cases}$$

Por tanto, las ecuaciones de los lenguajes de cada estado están dadas por la siguiente propiedad que es fácil de demostrar para cualquier $1 \leq i \leq n$:

$$L_i = \sum_{j=0}^{n} X_{i,j} L_j + Y_i$$

Dicha propiedad genera el llamado sistema de ecuaciones características de un AFN.

Finalmente, el Lema de Arden nos indica cómo calcular los conjuntos L_i. Esta será la idea a seguir para demostrar el Teorema de Análisis de Kleene:

1. Dado el autómata M construir los conjuntos $X_{i,j}$, Y_i.
2. Resolver el sistema de ecuaciones características mediante el Lema de Arden.
3. La solución para L_0 genera una expresión regular para $L(M)$.

1.3. Minimización de autómatas

Este proceso constructivo nos lleva a tener máquinas demasiado grandes, con $\boldsymbol{\varepsilon}$-transiciones. Por lo cual presentamos a continuación métodos para reducir el tamaño de autómatas. Estudiaremos la forma en que se pueden minimizar los autómatas para obtener máquinas más eficientes.

1.3.1. Eliminación de Estados Inaccesibles

Sea $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ un AFD. Decimos que un estado $q \in Q$ es accesible si y sólo si existe $w \in \Sigma^\star$ tal que $\delta^\star(q_0, w) = q$. Es decir, q es accesible si y sólo si el procesamiento de alguna cadena termina en el estado q.

El conjunto de estados accesibles de un autómata M se denota $Acc(M)$. Si un estado no es accesible decimos que es inaccesible.

Es claro que el conjunto $Acc(M)$ puede construirse de manera algorítmica, por ejemplo como sigue:

```
A_N := {q_0}  % estados accesibles
A_V := ∅      % estados verificados
while  A_N ≠ A_V do
  A_V := A_N
  A_N := A_N ∪ {q ∈ Q | δ(p, a) = q, a ∈ Σ, p ∈ A_N}
return A_N
```

Los estados inaccesibles en un autómata son inútiles y pueden ser eliminados sin afectar el lenguaje de aceptación como vemos a continuación:

Proposición 1 *Dado $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ un AFN, existe un AFD $M' = \langle Q', \Sigma, \delta', q_0, F'\rangle$ equivalente a M que contiene únicamente a los estados accesibles de M, es decir, $Q' = Acc(M)$ y por lo tanto no contiene estados inaccesibles. Para lo anterior basta definir M' como sigue:*

- $Q' = Acc(M)$
- $\delta' = \delta\restriction_{Q'}$
- $F' = F \cap Q'$

La prueba de la equivalencia $L(M) = L(M')$ es inmediata y se deja como ejercicio.

Debido a este resultado de ahora en adelante podemos suponer que un autómata no tiene estados inaccesibles.

1.3.2. Equivalencia de estados y el Autómata cociente

Puede ser el caso que unas partes de un autómata sean redundantes, es decir que las cadenas que son aceptadas por una parte del autómata también pueden ser procesadas y aceptadas por otra parte. Veamos cómo abstraer y generalizar partes de los autómatas para minimizar el número de estados sin afectar al lenguaje de aceptación.

Definición 3 *Decimos que dos estados $q, q' \in Q$ de un AFD son equivalentes $q \equiv q'$ si y sólo si:*

$$\forall w \in \Sigma^\star \ (\delta^\star(q, w) \in F \Leftrightarrow \delta^\star(q', w) \in F)$$

Es decir, si $\delta^\star(q, w), \delta^\star(q', w)$ son ambos finales o ambos no finales.

La relación $\equiv$ entre estados es una relación de equivalencia, es decir cumple lo siguiente:

- Reflexividad: $q \equiv q$.
- Simetria: si $q \equiv q'$ entonces $q' \equiv q$.
- Transitividad: si $q \equiv q'$ y $q' \equiv q''$ entonces $q \equiv q''$.

Adicionalmente la función de transición δ es compatible con $\equiv$, en el siguiente sentido:

$$\text{Si } q \equiv q' \text{ entonces } \forall a \in \Sigma \ \big(\delta(q, a) \equiv \delta(q', a)\big)$$

La relación de equivalencia $\equiv$ genera una **partición** del conjunto de estados dada por las clases de equivalencia de cada estado definidas como:

$$[q] := \{p \in Q \mid q \equiv p\}$$

Es decir, los conjuntos de estados $[q]$ cumplen lo siguiente:

- $\forall q \in Q \ ([q] \neq \varnothing\ )$.

- $\forall p, q \in Q$ ($[q] = [p]$ ó $[q] \cap [p] = \varnothing$).
- $\bigcup_{q \in Q} [q] = Q$.

Al agrupar por clases de equivalencia a los estados de un autómata, se puede calcular otro autómata llamado **autómata cociente** que tiene un número mínimo de estados.

Definición 4 *Dado un AFD $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ existe el autómata cociente $M/_{\equiv}$ también conocido como M^{min} que es la minimización de M y se define como $M^{min} = \langle Q_m, \Sigma, \delta_m, [q_0], F_m \rangle$ donde:*

- $Q_m := \{[q] \mid q \in Q\}$ *los estados son las clases de equivalencia*
- *la clase $[q_0]$ es el estado inicial.*
- $F_m := \{[q] \mid q \in F\}$
- $\delta_m : Q_m \times \Sigma \to Q_m$ *se define como* $\delta_m([q], a) = [\delta(q, a)]$

La definición anterior indica que dado un AFD $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ el autómata cociente $M/_{\equiv}$ es el autómata mínimo equivalente a M. Es decir, se tiene $L(M) = L(M/_{\equiv})$ y no existe un autómata equivalente a M con menos estados que $M/_{\equiv}$. La equivalencia entre M y M^{min} se sigue de la siguiente propiedad:

Lema 2 *Sean $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ un AFD y M^{min} su autómata cociente. Para cualesquiera $q \in Q$, $w \in \Sigma^\star$ se cumple $\delta^\star_m([q], w) = [\delta^\star(q, w)]$*
Demostración. La prueba es por inducción sobre w.

k-equivalencia La demostración anterior requiere de una relación de equivalencia que depende de la longitud de una cadena.
Definimos la relación de k-equivalencia para cualquier $k \in \mathbb{N}$ como sigue:

$$\forall w \in \Sigma^\star, |w| \leq k \to (\delta^\star(q, w) \in F \Leftrightarrow \delta^\star(q', w) \in F)$$

Es decir, para cualquier cadena w de longitud menor o igual que k, los estados $\delta^\star(q, w)$ y $\delta^\star(q', w)$ son ambos finales o ambos no finales.
Así $\equiv_k$ es una relación de equivalencia cuyas clases se denotan con $[q]_k$, es decir

$$[q]_k = \{p \in Q \mid q \equiv_k p\}$$

La relación de k-equivalencia cumple las siguientes propiedades:

P1 $q \equiv q'$ si y sólo si $\forall k \in \mathbb{N}(q \equiv_k q')$.

P2 $q \equiv_0 q'$ si y sólo si $q, q' \in F$ ó $q, q' \in Q - F$.

P3 $[q]_0 = F$ si y sólo si $q \in F$.

P4 Si $q \equiv_k q'$ entonces $q \equiv_{k-1} q'$.

P5 $[q]_k \subseteq [q]_{k-1}$

P6 Si $q \equiv_k q'$ entonces $\forall a \in \Sigma(\delta(q,a) \equiv_{k-1} \delta(q',a))$

P7 $q \equiv_k q'$ si y sólo si $q \equiv_{k-1} q'$ y $\forall a \in \Sigma(\delta(q,a) \equiv_{k-1} \delta(q',a))$

P8 Sea $P_k = \{[q]_k \mid q \in Q\}$ la partición dada por la relación $\equiv_k$ para cualquier $k \in \mathbb{N}$.
Si $P_k = P_{k-1}$ para alguna k entonces $P_k = P_m$ para toda $m \geq k$.

Con las definiciones anteriores podemos construir el autómata mínimo equivalente:

Definición 5 *Dado un AFD $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ el AFD mínimo asociado puede construirse como sigue:*

$Q := Acc(M)$ `% estados accesibles`
$P_0 := \{F, Q - F\}$ `% construir la particion inicial: estados finales y no-finales`
$k := 0$
`repeat {`
  $k := k + 1$
  $P_k := \{q \in P_{k-1} \mid \forall a \in \Sigma, [\delta(q,a)] = [\delta(q',a)]\}$
`} until` $P_k = P_{k-1}$
`return` P_k

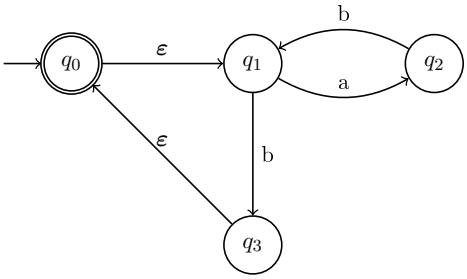
La partición P_k se construirá a partir de P_{k-1} manteniendo a dos estados q, q' en la misma clase si y sólo si para toda $a \in \Sigma$, los estados $\delta(q,a)$ y $\delta(q',a)$ estaban en la misma clase en P_{k-1}.
Es decir que P_k es la partición generada por $\equiv$: $P_k = Q/_{\equiv} = \{[q] \mid q \in Q\}$.

La defínción anterior está dada a través de un algoritmo el cual es correcto respecto a la especificación, ya que es consecuencia de la siguiente propiedad:

Proposición 2 *Si M es un AFD entonces la sucesión de particiones $P_0, P_1, \ldots, P_k$ generadas por las clases de k-equivalencia de estados se estaciona, es decir existe un $n \in \mathbb{N}$ tal que para toda $k \geq n$ se tiene que $P_k = P_n$. Más aún $n \leq |Q|$, es decir n es a lo más el número de estados de M.*

2. Ejemplo

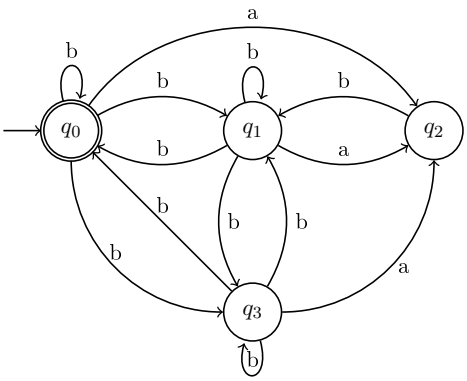
Sea M el siguiente autómata finito no-determinístico con transiciones $\boldsymbol{\varepsilon}$:



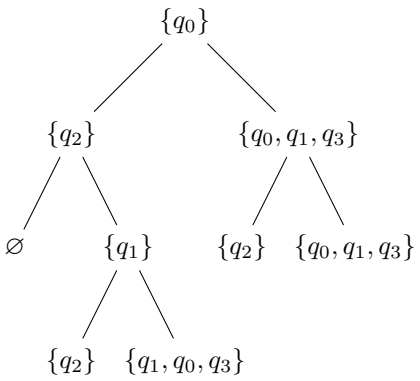
Usando los métodos descritos antes se obtendrá un autómata mínimo determinista de la siguiente forma:

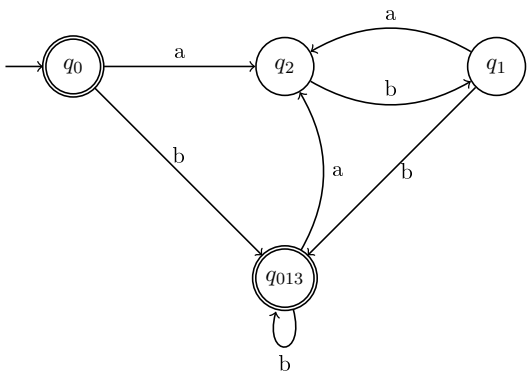
1. Se eliminan las ε-transiciones calculando los conjuntos Cl_ε de cada estado:

Q	Cl_ε	Q	a	b
q_0	$\{q_0, q_1\}$	q_0	$\{q_2\}$	$\{q_3, q_0, q_1\}$
q_1	$\{q_1\}$	q_1	$\{q_2\}$	$\{q_3, q_0, q_1\}$
q_2	$\{q_2\}$	q_2	$\varnothing$	$\{q_1\}$
q_3	$\{q_3, q_0, q_1\}$	q_3	$\{q_2\}$	$\{q_3, q_0, q_1\}$



2. Se transforma el autómata anterior en determinista:





3. Finalmente se minimiza:

	A		B	
	$\{q_0\}$	$\{q_0, q_1, q_3\}$	$\{q_2\}$	$\{q_1\}$
a	B	B	$--$	B
b	A	A	B	A

		A	B	C
	$\{q_0\}$	$\{q_0, q_1, q_3\}$	$\{q_2\}$	$\{q_1\}$
a	B	B	$--$	B
b	A	A	C	A