Autómatas finitos deterministas y no deterministas

Autómatas y Lenguajes Formales

13 de febrero de 2023

Autómata finito determinista

Me iré un poco rápido en estas notas pues varios conceptos los hemos visto en clase (en caso de quien esto lea no haya asistido a la clase puede consultar la bibliografía al final del documento).

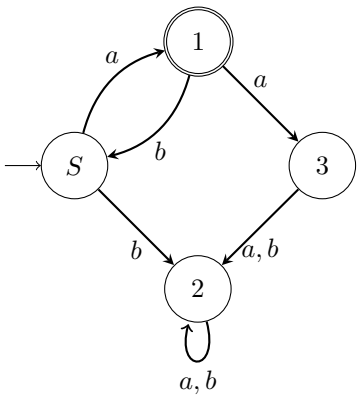
Definición 1 *Un autómata finito determinista es la quinteta*

$$M = (Q, \Sigma, \delta, s, F)$$

- Q *el conjunto finito de estados;*
- Σ *conjunto finito, alfabeto de entrada;*
- $\delta : Q \times \Sigma \to Q$ *la función de transición;*
- $s \in Q$ *es el estado inicial;*
- $F \subseteq Q$*, los estados de aceptación o estados finales.*

Veamos ejemplos para determinar que lenguaje acepta un autómata. Por ejemplo tenemos el siguiente autómata en diagrama:

¿Cómo analizamos para saber qué cadenas acepta, y por lo tanto que lenguaje? Por lo regular es por pura observación, pero esa observación se reduce a probar varias cadenas y a ver cuales pegan.



Lo primero que notamos es que el alfabeto es $\{a, b\}$, el conjunto de los estados es $\{S, 1, 2, 3\}$, el estado inicial es S y sólo hay un final que es 1. Propongo probar las cadenas: $\{a, b, ab, ba, aaa, bbb, aba, ababa\}$, a ver si podemos identificar algo. Para probar estas cadenas lo mejor es que ustedes en su pantalla o papel sigan con el dedo los estados a que lleva la lectura de cada carácter. Recuerden que lo único cercano a una memoria que tiene el autómata finito determinista son los distintos estados, así que la posición de su dedo guardará memoria de lo que ya se ha leído.

Para ejemplificar en papel ese seguimiento usaré la notación prestada de Lewis y Papadimitriou[3], la entrada es leída como (q, w) donde q es el estado donde se encuentra y w la subcadena que no se ha leído. Recuerden que la lectura se hace de izquierda a derecha, por convención. Las transiciones son marcadas por le símbolo $\vdash_A$ donde el subíndice A hace referencia al autómata especificado.

De esta forma leemos la primera cadena propuesta:

$$(S, a) \vdash_A (1, \epsilon)$$

Es decir, empezamos en el estado inicial S con la cadena a, empieza la actividad del autómata al marcar $\vdash_A$: leemos a y pasamos al estado 1. Como ya no hay más cadena la salida es marcada como $(1, \epsilon)$, ya estamos en el estado 1, que es final, y ya no hay nada más que leer, ϵ es la cadena vacía, es decir, nada.

Para el resto de cadenas: La cadena b

$$(S, b) \vdash_A (2, \epsilon).$$

Termina en el estado 2 que no es final, es decir que la cadena no es aceptada. La cadena ab

$$\begin{aligned}(S, ab) &\vdash_A (1, b)\\ (1, b) &\vdash_A (S, \epsilon)\end{aligned}$$

Termina en S que no es de aceptación, la cadena no es aceptada. Para ba

$$\begin{aligned}(S, ba) &\vdash_A (2, a)\\ (2, a) &\vdash_A (2, \epsilon)\end{aligned}$$

No es aceptada. Para aaa

$$\begin{aligned}(S, aaa) &\vdash_A (1, aa)\\ (1, aa) &\vdash_A (3, a)\\ (3, a) &\vdash_A (2, \epsilon)\end{aligned}$$

No es aceptada, lo mismo para bbb

$$\begin{aligned}(S, bbb) &\vdash_A (2, bb)\\ (2, bb) &\vdash_A (2, b)\\ (2, b) &\vdash_A (2, \epsilon)\end{aligned}$$

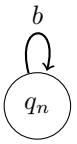


Figura 1: Ejemplo de como aceptar en un estado cualquier cantidad de b's o b^*. Sólo es un ejemplo, aún no está terminado el autómata.

Caso muy diferente con las últimas dos cadena, que sí son aceptadas: aba y $ababa$:

$$(S, aba) \vdash_A (1, ba)$$
$$(1, ba) \vdash_A (S, a)$$
$$(S, a) \vdash_A (1, \epsilon)$$

$$(S, ababa) \vdash_A (1, baba)$$
$$(1, baba) \vdash_A (S, aba)$$
$$(S, aba) \vdash_A (1, ba)$$
$$(1, ba) \vdash_A (S, a)$$
$$(S, a) \vdash_A (1, \epsilon)$$

Los dos terminan en el estado 1 que es de aceptación, ambas cadenas son aceptadas. Espero puedan ver de aquí una regularidad: se aceptan las cadenas que empiezan y terminan con a pero no puede haber más de una a ni una b consecutiva. No es el autómata más pequeño que hace esa labor, pero así no lo dieron.

Como les había mencionado un sitio útil para probar los autómatas de manera computacional es `http://automatonsimulator.com/`, ya su experiencia en la programación les dirá que lo más sano es dudar de que haga todo correctamente, revisen por su cuenta y validen si lo que hace el programa es correcto.

Otra parte importante a ejercitar es diseñar el autómata a partir de la descripción del lenguaje, por ejemplo:

- Partiendo del alfabeto $\{a, b\}$, construye el autómata que acepte las cadenas w tales que w tenga un número impar de a's y termine en b.

Los estados y las transiciones nos serán úriles para llevar cuenta de las a's y de la última b. Como las b's intermedias no nos interesan tanto ya supondrán que eso lo podemos poner como un lazo (*loop* como se le llama en inglés) sobre el estado. Esto se puede ver en la figura 1.

¿Cómo hacemos que lleve cuenta de que la cantidad de a's es impar? Puede parecer complicado pues números impares hay infinitos, pero hay un método más sencillo: al leer una a pasamos a un estado, el que lleva la cuenta de la imparidad, si se lee otra a se regresa al estado anterior, se lee la tercera a y se regresa al estado que lleva la cuenta, como se observa en la figura 2

Con estos ejemplos podemos darnos una idea de como construir el autómata completo, sólo n0os haría falta comprobar la última b, me parece eso no requerirá mayor explicación, espero así sea. En la figura 3 se puede ver el autómata completo.

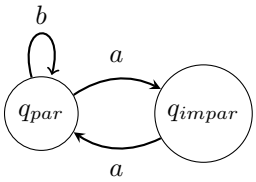


Figura 2: Ejemplo para identificar cantidades pares o impares de caracteres, sólo es un ejemplo no es un autómata terminado.

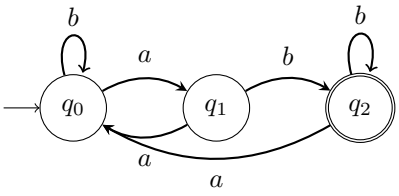


Figura 3: Autómata completo que acepta las cadenas w que contiene un número impar de a's y terminan en b.

Quizá requiera una explicación extra sobre lo que pasa si ya estando en q_2 y se lee una a, como ya llevábamos la cuenta de a impares, al leer una mas se tiene una cantidad par, regresamos a q_0 donde van las cantidades pares. Si la cadena termina ahí no es aceptada, de continuar se debe checar de nuevo que sea a impar y termine con b.

Autómatas finitos no deterministas

Definición 2 *Un autómata finito no determinista es la quinteta*

$$M = (Q, \Sigma, \Delta, S, F)$$

- Q *el conjunto finito de estados;*
- Σ *conjunto finito, alfabeto de entrada;*
- $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$ *la función de transición;*
- $S \subseteq Q$ *conjunto se estados iniciales;*
- $F \subseteq Q$*, los estados de aceptación o estados finales.*

Podemos ver un ejemplo sencillo. En la figura 4 tenemos un autómata no determinista de dos estados, con el alfabeto $\{a, b\}$.

De inspeccionar podemos ver que acepta las cadenas del tipo:

- La cadena vacía ϵ (el estado inicial es de aceptación).
- Las cadenas que son repeticiones de a $a, aa, aaa, aaaa, ...$ es decir a^* si incluimos aquí a la cadena vacía.

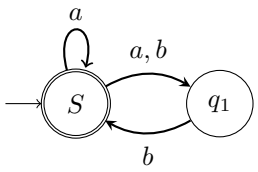


Figura 4: Ejemplo de autómata finito no determinista con alfabeto $\{a, b\}$ y dos estados.

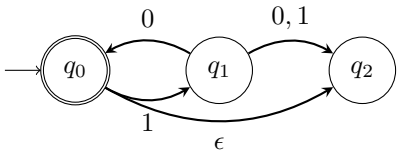


Figura 5: Ejemplo de autómata finito no determinista con alfabeto $\{0, 1\}$, tres estados y una transición ϵ.

- Las subcadenas con un número par de caracteres siempre y cuando el segundo carácter se b, $ab, bb, abbb, bbbb, ...$ Quizá ea más difícil de generalizar, yo lo pondría como $(ab)^+$ y $(bb)*$, líneas abajo veremos una forma más reducida de expresarlo.

¿Porqué es no determinista? Porque en es estado S hay dos posibles salidas para a, o se queda en S o pasa al estado q_1. Y en q_1 no hay transición para el carácter a. En este ejemplo hacen falta transiciones ϵ.

Un ejemplo con transiciones ϵ se puede ver en la figura 5. En este caso el autómata es no determinista porque hay estados que no tienen transición para ciertos caracteres (por ejemplo el q_2 que no tiene transición alguna de salida) y además hay una transición ϵ de q_0 a q_1.

Inspeccionando el autómata podemos ver que las cadenas que acepta son:

- La cadena vacía ϵ ya que el estado inicial es de aceptación, pero hay un camino (la transición ϵ al estado q_2) en el que no se acepta.

- todas las repeticiones ordenadas de 10, es decir $(10)^*$ incluyendo ya la cadena vacía, pero de igual forma esto sólo sucede en un camino posible, hay otro en el que no es aceptada

El resto de cadenas no son aceptadas. Las que consideramos que son aceptadas son aquellas en las que al menos en una posibilidad llegan a un estado de aceptación, pero hay posibilidades en las que no son aceptadas. Digamos que con que sea aceptada en uno basta para decir que son cadenas del lenguaje.

No muestro ejemplos de construcción pues son muy similares a las versiones deterministas salvo con algunas facilidades extras. En la mayoría de los casos es más fácil diseñar los autómatas como no deterministas.

Equivalencia entre autómatas finitos deterministas y no deterministas

Hemos dicho que ambos tipos de autómatas finitos son equivalentes, al menos en el lenguaje que aceptan. La manera como lo aceptan, los estados extras y donde quedan las cadenas rechazadas puede variar mucho, pero a final de cuentas lo que nos interesa es el lenguaje que aceptan. Para pasar de un autómata finito no determinista ($N(Q, \Sigma, \Delta, q_0, F)$)[1] a uno determinista ($A(Q', \Sigma, \delta, F')$) podemos seguir las siguientes instrucciones:

1. $Q' = \mathcal{P}(Q) = 2^Q$, los estados del autómata finito determinista serán los elementos del conjunto potencia del conjunto de estados del autómata no determinista original, o el conjunto de todos los subconjuntos de Q.

2. Para los estados del autómata determinista, $R \in Q'$ y un carácter del alfabeto $a \in \Sigma$, la transición para el determinista estará dada por

   $$\delta(R, a) = \{q \in Q | q \in \Delta(r, a) \text{ para alguna } r \in R\}.$$

   Como tenemos que los nuevos estados son elementos del conjunto potencia de los estados originales, existen estados $r \in Q$ tales que son elementos de un elemento $R \in Q'$. Una transición en el autómata no determinista N puede llevar a más de un estado, por ello en el autómata determinista A los llevamos a la unión de todos ellos:

   $$\delta(R, a) = \bigcup_{r \in R} \Delta(r, a)$$

3. $q'_0 = \{q_0\}$ por el momento dejamos el mismo estado inicial, pero no se acostumbren mucho a ello, va a cambiar un poco.

4. $F' = \{R \in Q' | R \text{ contiene un estado de aceptación de } N\}$, es decir, todos los estados del autómata determinista A que incluyen al menos uno de los estados finales del no determinista N son estados finales.

Y antes de aplicarlo hay que ver que se hace con las transiciones ϵ:

- Definimos el conjunto $E(R)$ que contiene a todos los estados a los que se puede llegar desde R a través de transiciones ϵ, incluido R

  $$E(R) = \{q | q \text{ se puede llegar desde } R \text{ viajando por } 0 \text{ o más transiciones } \epsilon\}.$$

- Modificamos las transiciones del autómata determinista A para marcar las transiciones posibles por ϵ, cambiando $\Delta(r, a)$ por $E(\Delta(r, a))$, es decir

  $$\delta(R, a) = \{q \in Q | q \in E(\Delta(r, a)) \text{ para algunas } r \in R\}$$

- Nuestro nuevo estado inicial cambiara de $q'_0 = \{q_0\}$ a $q'_0 = E(q_0)$, todos los estados que pueden alcanzarse por transición ϵ dese el estado inicial original q_0.

[1] Hay autómatas finitos deterministas con más de un estado inicial, en ese caso con el apoyo de un estado nuevo y transiciones ϵ se puede con relativa facilidad convertir a un autómata finito no determinista con un sólo estado inicial.

Para ver esto en acción hagamos un ejemplo. Convertiremos el autómata de la figura 5 a su versión determinista. Vamos por partes:

1. Los estado originales son $Q = \{q_0, q_1, q_2\}$, por lo tanto los estados del nuevo autómata determinista serán

$$Q' = \{\phi, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}.$$

2. Para ir viendo las transiciones hagamos una tabla:

Estado	0	1
ϕ	ϕ	ϕ
q_0^* (F)	ϕ	q_1
q_1	$q_0 \cup q_2$	q_2
q_2	ϕ	ϕ
$\{q_0, q_1\}$ (F)	$\phi \cup q_0 \cup q_2 = q_0 \cup q_2$	$q_1 \cup q_2$
$\{q_0, q_2\}$ (F)	ϕ	$q_1 \cup \phi = q_1$
$\{q_1, q_2\}$	$q_0 \cup q_2 \cup \phi = q_0 \cup q_2$	$q_2 \cup \phi = q_2$
$\{q_0, q_1, q_2\}$ (F)	$\phi \cup q_0 \cup q_2 \cup \phi = q_0 \cup q_2$	$q_1 \cup q_2 \cup \phi = q_1 \cup q_2$

3. q_0 es el estado inicial marcado en la tabla como q_0^*. Esto puede cambiar.

4. Los estados finales son $F = \{\{q_0\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_0, q_1, q_2\}\}$, todos los que incluyen a q_0. Están marcados como (F) en la tabla.

Hasta ahí vamos bien, pero nos faltan las transiciones ϵ, también las veremos por pasos como se indicó líneas arriba:

- Por suerte el único estado con transición ϵ es q_0 y sólo llega a q_2 (y al estado mismo), entonces

$$E(q_0) = \{q_0, q_2\} \tag{1}$$

- Ahora cambiamos las transiciones, todas las que llegaba al estado q_0 ahora llegan al estado $E(q_0)$

Estado	0	1
ϕ	ϕ	ϕ
q_0 (F)	ϕ	q_1
q_1	$E(q_0) \cup q_2 = \{q_0, q_2\} \cup q_2$	q_2
q_2	ϕ	ϕ
$\{q_0, q_1\}$ (F)	$E(q_0) \cup q_2 = \{q_0, q_2\} \cup q_2$	$q_1 \cup q_2$
$\{q_0, q_2\}^*$ (F)	ϕ	q_1
$\{q_1, q_2\}$	$E(q_0) \cup q_2 = \{q_0, q_2\} \cup q_2$	q_2
$\{q_0, q_1, q_2\}$ (F)	$E(q_0) \cup q_2 = \{q_0, q_2\} \cup q_2$	$q_1 \cup q_2$

- Ahora debemos ver si cambia el estado inicial, y coincide en que justo el estado inicial tiene una transición ϵ. Entonces el estado inicial cambia de q_0 a $E(q_0) = \{q_0, q_2\}$, información ya incluida en la tabla de líneas arriba.

Y listo, tenemos nuestro autómata finito determinista A equivalente al finito no determinista N. Con eso bastaría, pero en caso de que sean más visuales de esa tabla de transiciones podemos armar el diagrama, pero eso lo agrego después, pueden hacerlo ustedes y me pasan como queda y lo agrego.

Expresiones regulares

Definición 3 *Decimos que R es una expresión regular si R es*

1. *a para alguna a en el alfabeto Σ ($\{a\}$),*
2. *ϵ la cadena vacía ($\{\epsilon\}$),*
3. *ϕ (el lenguaje vacío),*
4. *$(R_1 \cup R_2)$ o $(R_1 + R_2)$, donde R_1 y R_2 son expresiones regulares (unión de lenguajes),*
5. *$(R_1 \cdot R_2) = (R_1 R_2)$ donde R_1 y R_2 son expresiones regulares (concatenación de lenguajes), o*
6. *(R_1^*) con R_1 una expresión regular (estrella de Kleene del lenguaje).*

De estas expresiones regulares podemos construir autómatas finitos no deterministas (por facilidad y hacerlo más directo, con un poco de análisis se puede hacer determinista), que pueden ser usados como células para armar expresiones más complicadas.

Por ejemplo para la expresión regular a tenemos el autómata que se muestra en la figura 6

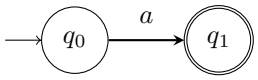


Figura 6: Autómata que acepta el lenguaje determinado por la expresión regular a.

De aquí se figurarán que para la expresión regular ϵ basta con hacer el estado inicial final y no incluir transición alguna. Para el lenguaje regular definido por ϕ (el lenguaje vacío) se tiene un estado inicial sin transiciones y sin ningún estado final, el lenguaje aceptado es vacío, ni la cadena vacía acepta.

Quizá más interesante sea ver como construir el autómata para la unión, concatenación y estrella de Kleene de expresiones regulares. Para la unión de expresiones realizamos la unión de los autómatas que aceptan cada expresión, por ejemplo el término $(a + b)$ o $(a \cup b)$ se puede ver en la figura 7

Se crea un nuevo estado inicial que sea para ambos autómatas y se pegan a través de transiciones ϵ a los antiguos estados iniciales de ambos autómatas. Los estados finales son los mismos.

Para el caso de la concatenación se usa igual transiciones ϵ como pegamento, solo que en este caso es de izquierda a derecha. Por ejemplo para la expresión ab podemos ver en la figura 8 como se hace el "pegado" de los autómatas que aceptan los lenguajes de las expresiones a y b.

El siguiente caso por ver sería la estrella de Kleene de una expresión regular, si partimos del caso de la concatenación, por ejemplo $(ab)^*$, el autómata resultante sería el que se muestra en la figura 9. Para hacer la estrella de Kleene basta con hacer una transición ϵ. Para dejarlo de la forma más general, para no regresar al estado inicial se puede agregar un estado. Puede estar de más, o puede ser útil después, por si las moscas lo agregamos.

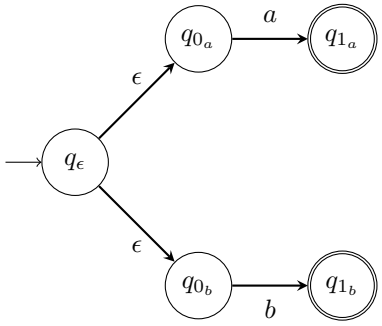


Figura 7: Autómata que acepta el lenguaje determinado por la expresión regular $a + b$.

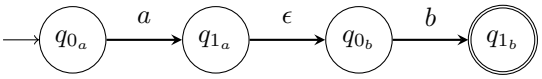


Figura 8: Autómata que acepta el lenguaje determinado por la expresión regular ab.

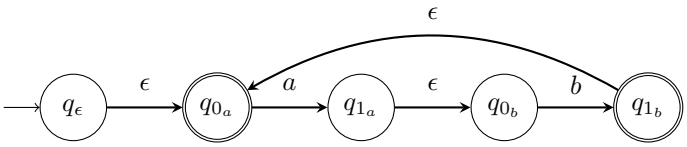


Figura 9: Autómata que acepta el lenguaje determinado por la expresión regular $(ab)^*$.

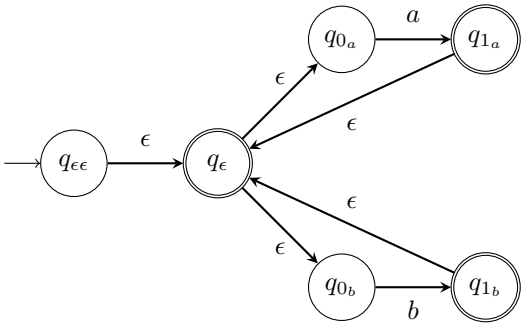


Figura 10: Autómata que acepta el lenguaje determinado por la expresión regular $(a \cup b)^*$.

Para el caso $(a \cup b)^*$ cambia un poco, pero se sigue la misma idea de las transiciones ϵ para adherir y hacer la estrella de Kleene, como se muestra en la figura 10.

A partir de estas formas básicas se pueden construir los autómatas equivalentes a las expresiones regulares (equivalentes ya que aceptan el mismo lenguaje).

Referencias

[1] Kozen, Dexter C. “Automata and Computability” Springer (1997)

[2] Sipser, Michael “Introduction to the Theory of Computation” 2a ed., Thomson Course Tecnology (2006)

[3] Lewis, Harry R. y Papadimitriou, Christos H. “Elements of the Theory of Computation”2a ed., Prentice-Hall Inc. (1998).