LENGUAJES REGULARES

Abril de 2023

Introducción

En la clase de hoy, empezamos la parte de curso sobre teoría de autómatas.

Los autómatas son modelos computacionales que se utilizan en el diseño de software para sistemas informáticos.

Entre sus principales aplicaciones, podemos mencionar las siguientes:

Descripción básica

(1) Diseño de analizadores léxicos y procesadores de textos.

(2) Diseño de programas para buscar palabras o frases en textos amplios.

(3) Diseño de software para crear protocolos de comunicación o protocolos de intercambio de información.

(4) Diseño de compiladores.

Descripción básica

Los autómatas son reconocedores de lenguajes.

Dicho de manera intuitiva, un autómata M para un lenguaje L es un modelo computacional que recibe como entrada una tira de símbolos x y entonces determina si x pertenece o no al lenguaje L.

Hay diversos tipos de autómatas que se utilizan en el diseño de software de sistemas. En esta parte del curso, estudiaremos los llamados autómatas finitos y autómatas con pila, que son los tipos de autómatas que se utilizan en el diseño de compiladores.

Antes de definir el concepto de autómata, necesitamos introducir algunos conceptos básicos sobre lenguajes formales, que veremos en la clase de hoy.

Alfabetos y palabras

Un alfabeto es un conjunto no vacío de símbolos.

Una palabra sobre un alfabeto Σ es una secuencia finita de símbolos de Σ.

Denotamos por λ a la palabra vacía, es decir, a la palabra que no tiene ningún símbolo.

La longitud de una palabra x es el número de símbolos que la componen, contando repeticiones. Denotamos a la longitud de una palabra x por $|x|$.

Por ejemplo, consideremos las palabras $x = 1024$ y $z = 1509450753$ sobre el alfabeto $\{0, 1, \ldots, 9\}$. Entonces, $|x| = 4$ y $|z| = 10$.

Alfabetos y palabras

Si x es una palabra sobre un alfabeto Σ y a es un símbolo de Σ, denotamos por $n_a(x)$ al número de apariciones del símbolo a en x.

Por ejemplo, si consideramos la palabra $z = 1509450753$ sobre el alfabeto $\{0, 1, \ldots, 9\}$, tenemos que
$n_0(z) = 2, n_1(z) = 1, n_2(z) = 0, n_3(z) = 1, n_4(z) = 1, n_5(z) = 3,$
$n_6(z) = 0, n_7(z) = 1, n_8(z) = 0, n_9(z) = 1.$

Si Σ es un alfabeto, denotamos por Σ^* al conjunto de todas las palabras sobre Σ.

El concepto de lenguaje

Un lenguaje es un conjunto de palabras sobre un alfabeto.

Ejemplos de lenguajes:

(1) El conjunto $\{x \in \{a, b, c\}^* : n_a(x) = n_b(x) = n_c(x)\}$.

(2) El conjunto de identificadores de un lenguaje de programación.

En el caso del lenguaje Java, el lenguaje de los identificadores está definido en el alfabeto
$\Sigma = \{0, \ldots, 9\} \cup \{a, b, \ldots, z\} \cup \{A, B, \ldots, Z\} \cup \{\_, \$\}$.

(3) El conjunto de programas de un lenguaje de programación.

Este lenguaje está definido en el alfabeto
$\Sigma = \{0, \ldots, 9\} \cup \{a, b, \ldots, z\} \cup \{A, B, \ldots, Z\} \cup \{$ símbolos de puntuación del lenguaje de programación $\} \cup \{$ palabras reservadas del lenguaje de programación $\}$.

Operaciones básicas para palabras

La operación más importante es la concatenación, que consiste en la yuxtaposición de una palabra x con una palabra y, que se representa por $x \cdot y$, o más abreviadamente por xy. Por tanto, la palabra $x \cdot y$ consiste en poner los símbolos de x y a continuación los símbolos de y.

Por ejemplo, si $x = 100$ e $y = 01$, tenemos que $x \cdot y = xy = 10001$ e $y \cdot x = yx = 01100$.

Utilizaremos la notación exponencial para representar la concatenación repetida de una palabra con ella misma. Si x es una palabra, definimos

$$x^0 = \lambda,$$
$$x^{n+1} = x^n \cdot x.$$

Por ejemplo, si $x = 011$, tenemos que $x^0 = \lambda$, $x^1 = x = 011$, $x^2 = 011011$, $x^3 = 011011011$, ...

Operaciones básicas para palabras

Otra operación importante es la inversa de una palabra x. Si $x = a_1 \dots a_n$, definimos la inversa de x por la palabra $x^I = a_n \dots a_1$.

Recordemos que λ representa la palabra vacía, es decir, la palabra que no tiene ningún símbolo. Tenemos entonces:

(1) Para toda palabra x, se tiene que $x \cdot \lambda = \lambda \cdot x = x$.

(2) Para cualesquiera palabras x, y, se tiene que $(x \cdot y)^I = y^I \cdot x^I$.

Operaciones básicas para lenguajes

(1) La unión.

Si L_1 y L_2 son dos lenguajes sobre un alfabeto Σ, entonces $L_1 \cup L_2 = \{x \in \Sigma^* : x \in L_1 \text{ o } x \in L_2\}$.

(2) La intersección.

Si L_1 y L_2 son dos lenguajes sobre un alfabeto Σ, entonces $L_1 \cap L_2 = \{x \in \Sigma^* : x \in L_1 \text{ y } x \in L_2\}$.

(3) La complementación.

Si L es un lenguaje sobre un alfabeto Σ, definimos el complementario de L por $\overline{L} = \{x \in \Sigma^* : x \notin L\}$.

(4) La diferencia.

Si L_1 y L_2 son dos lenguajes, definimos el lenguaje $L_1 \setminus L_2 = \{x \in L_1 : x \notin L_2\}$.

Operaciones básicas para lenguajes

(5) La concatenación.

Si L_1 y L_2 son dos lenguajes, definimos la concatenación de L_1 con L_2 por $L_1 \cdot L_2 = L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$.

Por ejemplo, si $L_1 = \{0, 01, 100\}$ y $L_2 = \{01, 10\}$, tenemos que $L_1 L_2 = \{001, 010, 0101, 0110, 10001, 10010\}$ y $L_2 L_1 = \{010, 0101, 01100, 100, 1001, 10100\}$.

Utilizaremos la notación exponencial para representar la concatenación repetida de un lenguaje consigo mismo. Si L es un lenguaje, definimos:

$$L^0 = \{\lambda\},$$
$$L^{n+1} = L^n \cdot L.$$

Por ejemplo, si $L = \{1, 00\}$, tenemos que $L^0 = \{\lambda\}$, $L^1 = L = \{1, 00\}$, $L^2 = L \cdot L = \{11, 100, 001, 0000\}$, $L^3 = L^2 \cdot L = \{111, 1100, 1001, 10000, 0011, 00100, 00001, 000000\}$, ...

Operaciones básicas para lenguajes

(6) La clausura de un lenguaje.

Si L es un lenguaje, definimos la clausura de L por $L^* = \{x_1 x_2 \dots x_n : n \geq 0, x_1, x_2, \dots, x_n \in L\}$. Es decir, L^* es el lenguaje formado por todas las posibles concatenaciones de palabras de L.

Se tiene que $L^* = \bigcup\{L^n : n \geq 0\}$.

Por ejemplo, consideremos $\Sigma = \{0, 1\}$. Si tomamos $L = \{0\}$, tenemos que $L^* = \{0^n : n \geq 0\}$.

Y si tomamos $L = \{00, 1\}$, entonces $L^* = \{x \in \{0, 1\}^* :$ en x todos los ceros aparecen en pares adyacentes$\}$.

Lenguajes regulares

Hay muchos lenguajes que admiten descripciones que utilizan únicamente los símbolos de un alfabeto y las operaciones básicas entre lenguajes. Son los llamados lenguajes regulares, los cuales se describen por las llamadas expresiones regulares, que definimos a continuación.

Definición de expresión regular

Una expresión regular sobre un alfabeto Σ es una palabra sobre el alfabeto $\Sigma \cup \{(,), \emptyset, \lambda, \cup, \cdot, *\}$ generada por las siguientes reglas:

(1) $\emptyset$ y λ son expresiones regulares.

(2) Para todo $a \in \Sigma$, a es una expresión regular.

(3) Si α y β son expresiones regulares, también lo son $(\alpha \cup \beta)$ y $(\alpha \cdot \beta)$.

(4) Si α es una expresión regular, también lo es α^*.

Lenguaje asociado a una expresión regular

Si α es una expresión regular, definimos el lenguaje $L(\alpha)$ asociado a α por las siguientes reglas:

(1) Definimos $L(\emptyset) = \emptyset$ y $L(\lambda) = \{\lambda\}$.

(2) Si $a \in \Sigma$, definimos $L(a) = \{a\}$.

(3) Definimos $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$.

(4) Definimos $L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$.

(5) Definimos $L(\alpha^*) = L(\alpha)^*$.

Se dice entonces que un lenguaje L es regular, si hay una expresión regular α tal que $L = L(\alpha)$.

En ocasiones, para simplificar la notación, escribiremos α en lugar de $L(\alpha)$.

Ejemplos

(1) Si $\alpha = (0 \cup 1)^*$, $L(\alpha) = \{0,1\}^*$, es decir, es el lenguaje de todas las palabras de bits, ya que aplicando la definición anterior tenemos:
$L((0 \cup 1)^*) = (L(0 \cup 1))^* = (L(0) \cup L(1))^* = (\{0\} \cup \{1\})^* = \{0,1\}^*$.

(2) Si $\alpha = (0 \cup 1)^* 1$, $L(\alpha) = \{x \in \{0,1\}^* : x$ acaba en $1\}$.

(3) Si $\alpha = ((0 \cup 1) \cdot (0 \cup 1) \cdot (0 \cup 1))^*$, entonces $L(\alpha)$ es el conjunto de las palabras de bits cuya longitud es un múltiplo de 3.

(4) Si $\alpha = (1 \cup 2 \cup 0(1 \cup 2)^* 0)^*$, entonces
$L(\alpha) = \{x \in \{0,1,2\}^* : x$ tiene un número par de ceros $\}$.

Ejemplos

(5) Es bien sabido que todos los tipos de datos de los lenguajes de programación se pueden representar mediante expresiones regulares. Por ejemplo, si $\alpha = (+ \cup - \cup \lambda) \cdot (0 \cup 1 \cup \cdots \cup 9) \cdot (0 \cup 1 \cup \cdots \cup 9)^*$, entonces $L(\alpha)$ es el tipo entero.

Expresiones regulares equivalentes

Dos expresiones regulares α, β son equivalentes, si $L(\alpha) = L(\beta)$. Escribiremos entonces $\alpha \equiv \beta$.

Veamos algunos ejemplos:

(1) Las expresiones regulares $\alpha = 0^*1^*$ y $\beta = (01)^*$ no son equivalentes, ya que por ejemplo la palabra $0101 \in L(\beta)$ pero $0101 \notin L(\alpha)$, ya que las palabras de $L(\alpha)$ están formadas por un bloque de ceros seguido de un bloque de unos, por lo que es imposible que en una palabra de $L(\alpha)$ aparezca un 1 antes que un 0.

Expresiones regulares equivalentes

(2) Veamos ahora un ejemplo de dos expresiones regulares distintas que son equivalentes. Consideremos $\alpha = (0 \cup 1)^*$ y $\beta = (0^*1)^*0^*$. Como antes hemos visto, tenemos que $L((0 \cup 1)^*) = \{0, 1\}^*$, es decir, es el lenguaje de todas las palabras de bits. Por tanto, $L(\beta) \subseteq L(\alpha)$. Demostramos ahora que $L(\alpha) \subseteq L(\beta)$. Para ello, observamos que toda palabra x de $\{0, 1\}^*$ se puede representar agrupando cada 1 que aparece en la palabra con los ceros que le preceden; cada uno de estos grupos es una palabra de 0^*1. Por tanto, todos los grupos juntos forman una palabra $(0^*1)^*$. Además, hay que añadir los ceros que eventualmente pueda haber al final de la palabra x y que no vayan seguidos de ningún uno, los cuales pertenecen a 0^*. Por tanto, $x \in L(\beta)$.

Autómatas finitos

Los lenguajes regulares se pueden representar mediante los llamados autómatas finitos, que son los autómatas más simples.

Un autómata finito tiene asociada una cinta de entrada, la cual es una cinta de lectura que está dividida en celdas. La información de la cinta se encuentra entonces almacenada en las celdas, y el autómata accede a dicha información mediante un puntero que puede leer en un instante dado el contenido de una celda de la cinta. Al producirse un paso de cómputo, el puntero se mueve a la siguiente celda a la derecha en la cinta de entrada.

El autómata tiene además asociada una "unidad de control", que en un paso de cómputo se encuentra en un cierto estado. Al pasar entonces al siguiente paso de cómputo, el estado puede variar.

Definición de autómata determinista

Un autómata determinista es una estructura $M = (K, \Sigma, \delta, q_0, F)$ donde:

(1) K es un conjunto finito y no vacío de estados.

(2) Σ es un alfabeto finito, el alfabeto de entrada.

(3) δ es una función de $K \times \Sigma$ en K, a la que se denomina función de transición.

(4) $q_0 \in K$ es el estado inicial.

(5) $F \subseteq K$ es el conjunto de estados aceptadores (o estados finales).

Observaciones

Llamaremos símbolo actual al carácter accesible a través del puntero en un instante dado.

Inicialmente, el autómata se encuentra en el estado inicial q_0 con una entrada $x \in \Sigma^*$ escrita al comienzo de la cinta (de manera que el puntero señala a la primera celda).

Los cómputos del autómata se realizan por medio de la función δ. Es decir, para pasar de un paso de cómputo al siguiente se aplica la función δ. El argumento de δ es el par (q, a) donde q es el estado actual del autómata y a es el símbolo actual. La imagen de la función δ es un estado p, que corresponde al estado del autómata en el siguiente paso de cómputo.

Representación de un autómata mediante un grafo

Consideremos un autómata determinista $M = (K, \Sigma, \delta, q_0, F)$. Para representar M mediante un grafo, seguimos el siguiente algoritmo:

(1) Los nodos del grafo son los estados del autómata.

(2) Se marca el estado inicial con una flecha, y se marca cada estado aceptador con un doble círculo o con una cruz.

(3) Si $\delta(q, a) = p$, se dibuja un arco en el grafo de q a p con etiqueta a.

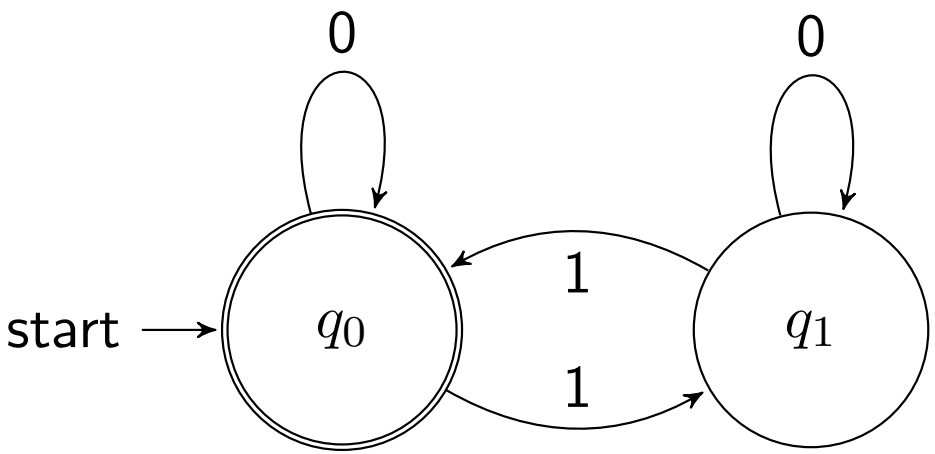
Ejemplo

Consideremos el autómata determinista $M = (K, \Sigma, \delta, q_0, F)$ donde $K = \{q_0, q_1\}$, $\Sigma = \{0, 1\}$, $F = \{q_0\}$ y δ está definida por la siguiente tabla:

q	σ	$\delta(q, \sigma)$
q_0	0	q_0
q_0	1	q_1
q_1	0	q_1
q_1	1	q_0

Grafo del autómata

El autómata anterior se puede representar mediante el siguiente grafo:



Nociones básicas para autómatas deterministas

Si $M = (K, \Sigma, \delta, q_0, F)$ es un autómata determinista, definimos una configuración de M como una palabra $px \in K\Sigma^*$.

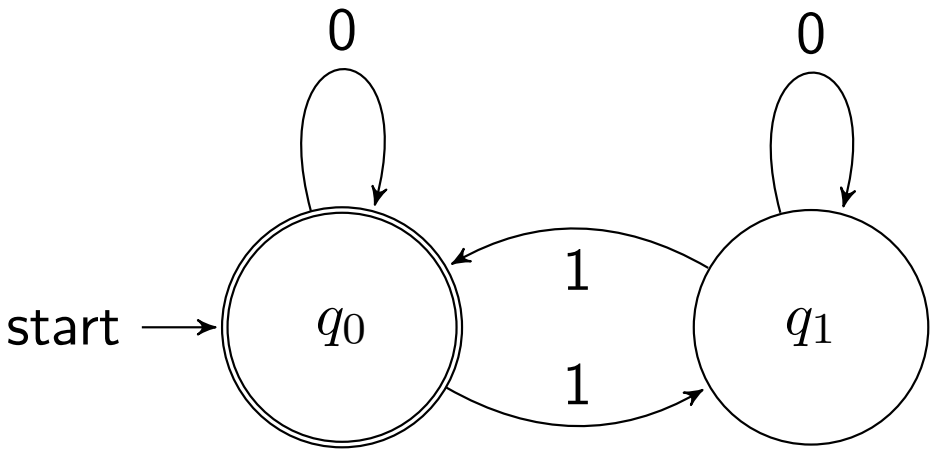
Si en un paso de cómputo la configuración de M es px, esto significa que el autómata M se encuentra en el estado p y x es la parte de la palabra de entrada que todavía no se ha leído.

Si px y qy son configuraciones de M, decimos que px produce qy en un paso de cómputo, lo que representamos por $px \vdash_M qy$, si en M podemos pasar de px a qy aplicando una vez la función de transición.

Y decimos que px produce qy, lo que representamos por $px \vdash_M^* qy$, si en M podemos pasar de px a qy aplicando un número finito de veces la función de transición.

Ejemplo

Consideremos el autómata M del ejemplo anterior:



Tenemos entonces el siguiente cómputo para la entrada $x = 00110$:

$$q_0 00110 \vdash_M q_0 0110 \vdash_M q_0 110 \vdash_M q_1 10 \vdash_M q_0 0 \vdash_M q_0.$$

Lenguaje asociado a un autómata determinista

Sea $M = (K, \Sigma, \delta, q_0, F)$ un autómata determinista.

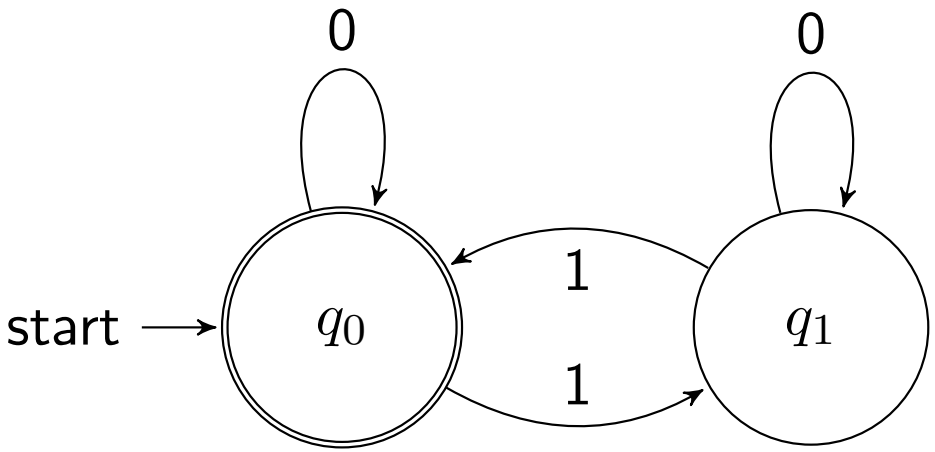
(a) Una palabra $x \in \Sigma^*$ es reconocida (o aceptada) por M, si existe un estado $q \in F$ tal que $q_0 x \vdash_M^* q$.

(b) Definimos el lenguaje reconocido por M por

$$L(M) = \{x \in \Sigma^* : x \text{ es reconocida por } M\}.$$

Ejemplo 1

Consideremos el autómata M visto anteriormente:

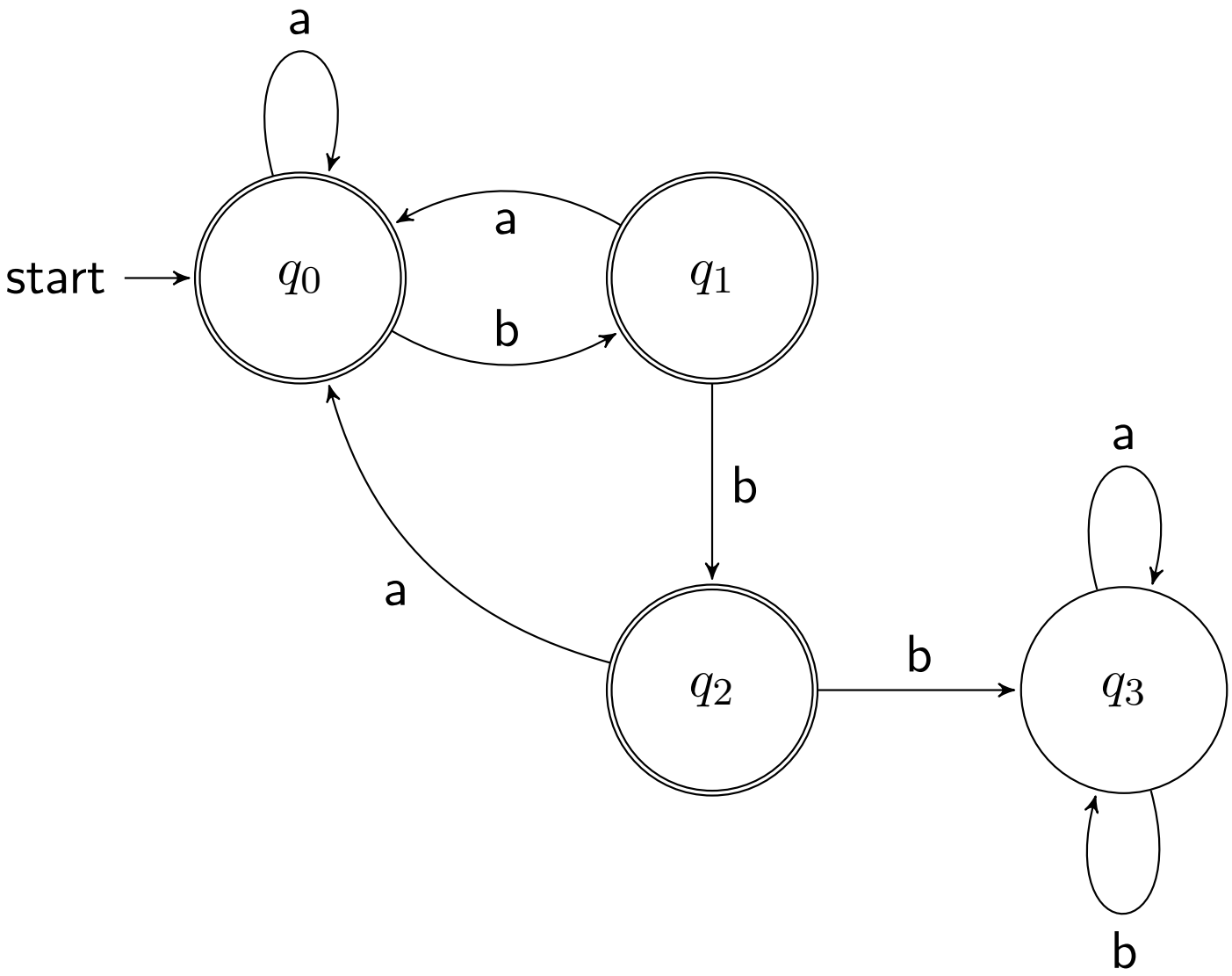


Observamos que si en un paso de cómputo del autómata estamos en el estado q_0, entonces el número de unos leídos en la entrada es par; y si estamos en el estado q_1, entonces el número de unos leídos en la entrada es impar. Así pues, como q_0 es el estado aceptador, tenemos que

$$L(M) = \{x \in \{0,1\}^* : n_1(x) \text{ es par}\}.$$

Ejemplo 2

M':



Ejemplo 2

Observamos que si entran tres b's seguidas, el autómata va al estado q_3, el cual no es aceptador y del cual ya no saldremos, ya que tanto si entra una a como una b seguiremos en q_3. Y si no entran tres b's seguidas, el autómata terminará el cómputo o bien en el estado q_0, o en el q_1, o en el q_2, los cuales son estados aceptadores. Por tanto,

$L(M') = \{x \in \{a, b\}^* : \text{en } x \text{ no aparecen tres b's consecutivas}\}$.

Equivalencia entre expresiones regulares y autómatas deterministas

La equivalencia viene dada por el siguiente teorema.

Teorema

Para todo lenguaje L, existe una expresión regular α tal que $L = L(\alpha)$ si y sólo si existe un autómata determinista M tal que $L(M) = L$.

Este teorema expresa entonces la equivalencia entre el concepto de expresión regular y el concepto de autómata determinista.

Programación de autómatas deterministas

Los autómatas deterministas son especificaciones de programas. Para escribir un programa en Java o en C correspondiente a un autómata determinista, podemos proceder de la siguiente manera:

(1) Representamos a los estados del autómata por números naturales, representando por el 0 al estado inicial.

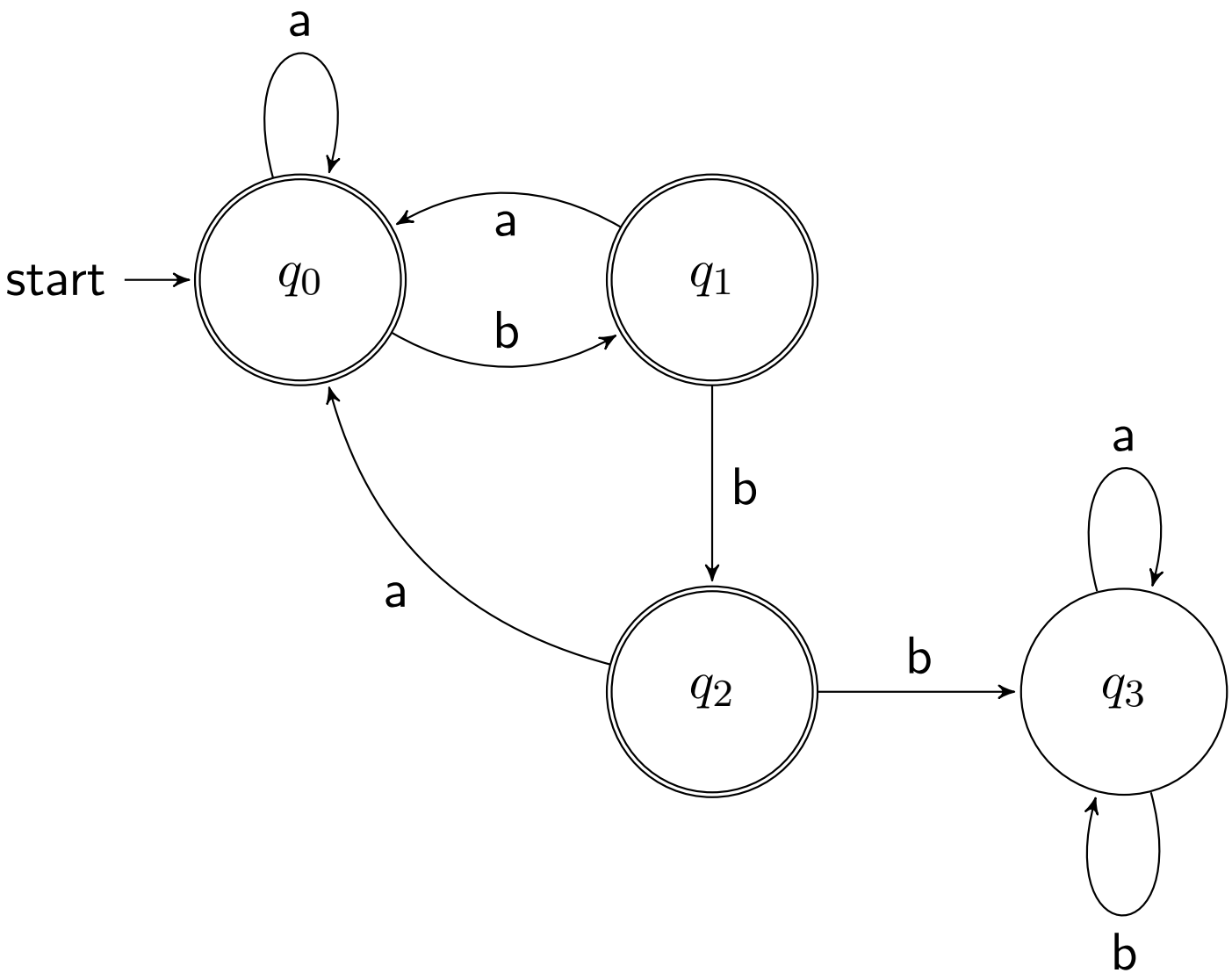
(2) Representamos a los símbolos del alfabeto de entrada por caracteres. Y representamos por el carácter $ el final de la palabra de entrada.

(3) Representamos el cálculo del autómata mediante un bucle "while", en el que describimos mediante una instrucción switch-case las transiciones del autómata en las que cambia el estado.

(4) Al salir del bucle "while", comprobamos si el estado en el que estamos es un estado aceptador.

Ejemplo

Consideremos el autómata determinista M', que vimos en la última clase:



Ejemplo

Representamos a cada estado q_i por i. Podemos escribir entonces el siguiente método en Java para simular el autómata M':

```
public boolean simular (String entrada)
{ int q = 0, i = 0;
  char c = entrada.charAt(0);
  while (c != '$')
  { switch(q)
        { case 0:
        if (c == 'b') q = 1;
        break;
        case 1:
        if (c == 'a') q = 0; else if (c == 'b') q = 2;
        break;
        case 2:
        if (c == 'a') q = 0; else return false;
        break;}
  c = entrada.charAt(++i); } return true; }
```