Problema 1. Definir autómatas deterministas que reconozcan los siguientes lenguajes.

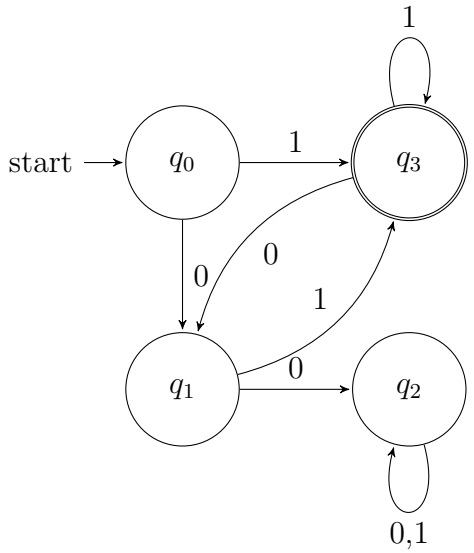
(a) $\{x \in \{0,1\}^* : x$ acaba en 1 y no contiene 00$\}$.

(b) $\{x \in \{0,1\}^* : x$ no contiene ni 00 ni 11$\}$.

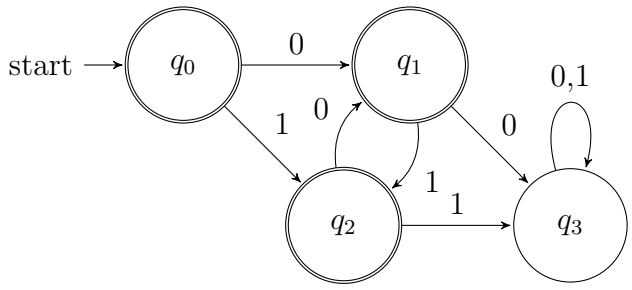
(c) $\{x \in \{0,1\}^* : x$ contiene 01 o 110 $\}$

Solución:

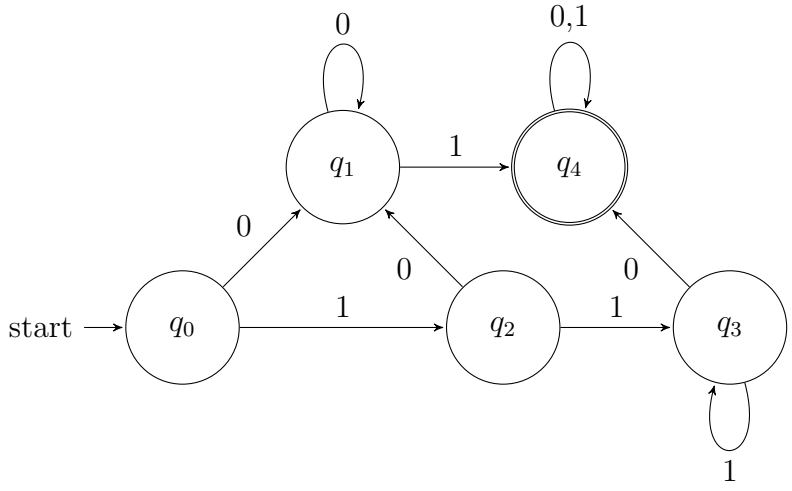
(a) Definimos el siguiente autómata determinista, que consta de los estados q_0, q_1, q_2 y q_3, donde q_0 es el estado inicial y q_3 es el único estado aceptador.



(b) Definimos el siguiente autómata determinista, que consta de los estados q_0, q_1, q_2 y q_3, donde q_0 es el estado inicial y q_0, q_1 y q_2 son los estados aceptadores.



(c) Definimos el siguiente autómata determinista, que consta de los estados q_0, q_1, q_2, q_3 y q_4, donde q_0 es el estado inicial y q_4 es el único estado aceptador.

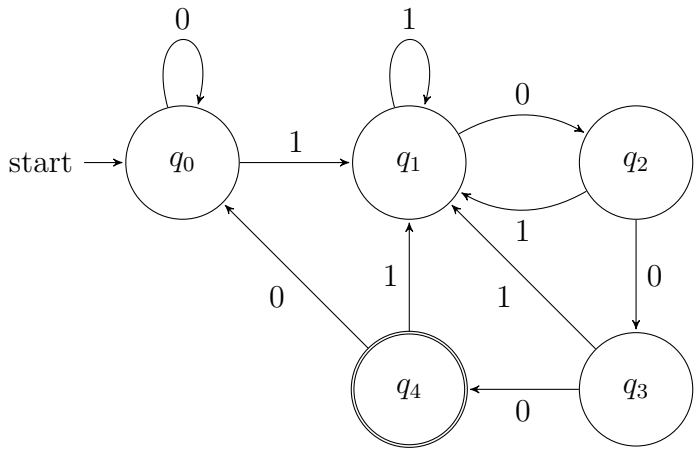


Problema 2. (a) Definir un autómata determinista que reconozca el lenguaje de las palabras de bits que acaban en 1000.

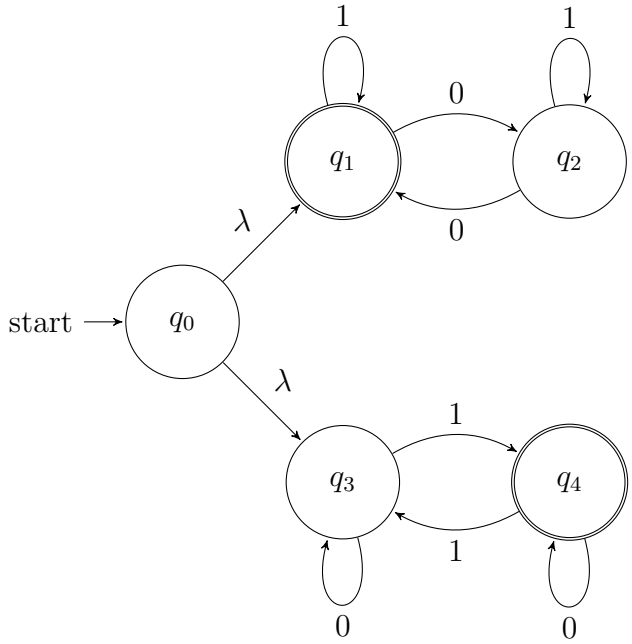
(b) Definir un autómata indeterminista que reconozca el lenguaje $\{x \in \{0,1\}^* : n_0(x)$ es par ó $n_1(x)$ es impar $\}$.

Solución:

(a) Definimos el autómata por el siguiente gráfico, donde q_4 es el único estado aceptador.



(b) Definimos el autómata por el siguiente gráfico, donde q_1 y q_4 son los estados aceptadores.

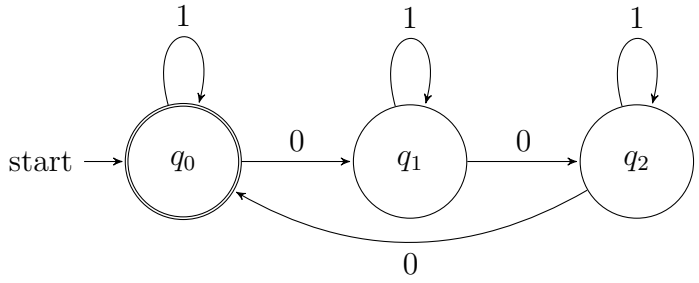


Problema 3. (a) Definir un autómata determinista M tal que $L(M) = \{x \in \{0,1\}^* : n_0(x) \text{ es un múltiplo de } 3\}$.

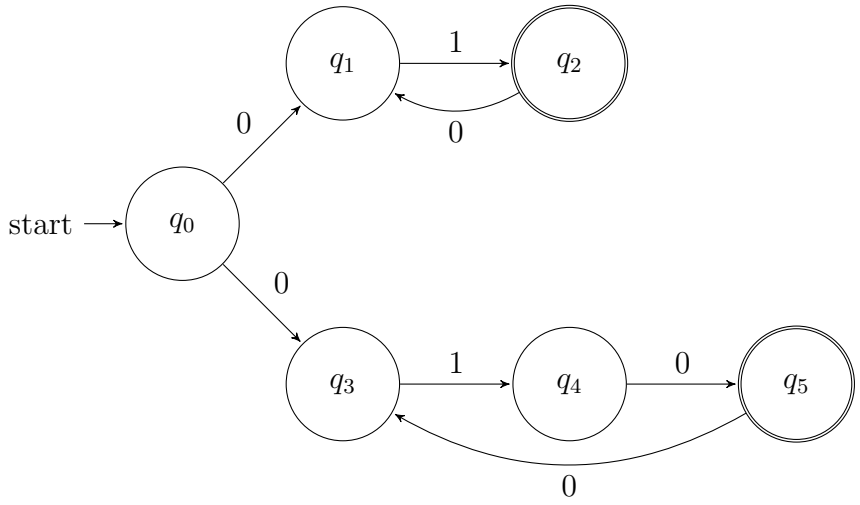
(b) Definir un autómata indeterminista M tal que $L(M) = \{(01)^n : n \geq 1\} \cup \{(010)^n : n \geq 1\}$, es decir, $L(M)$ es el lenguaje de las palabras formadas por 01 repetido una o más veces, o por 010 repetido una o más veces.

Solución:

(a) Definimos el siguiente autómata determinista, que consta de los estados q_0, q_1 y q_2, donde q_0 es el estado inicial y es asimismo el único estado aceptador.



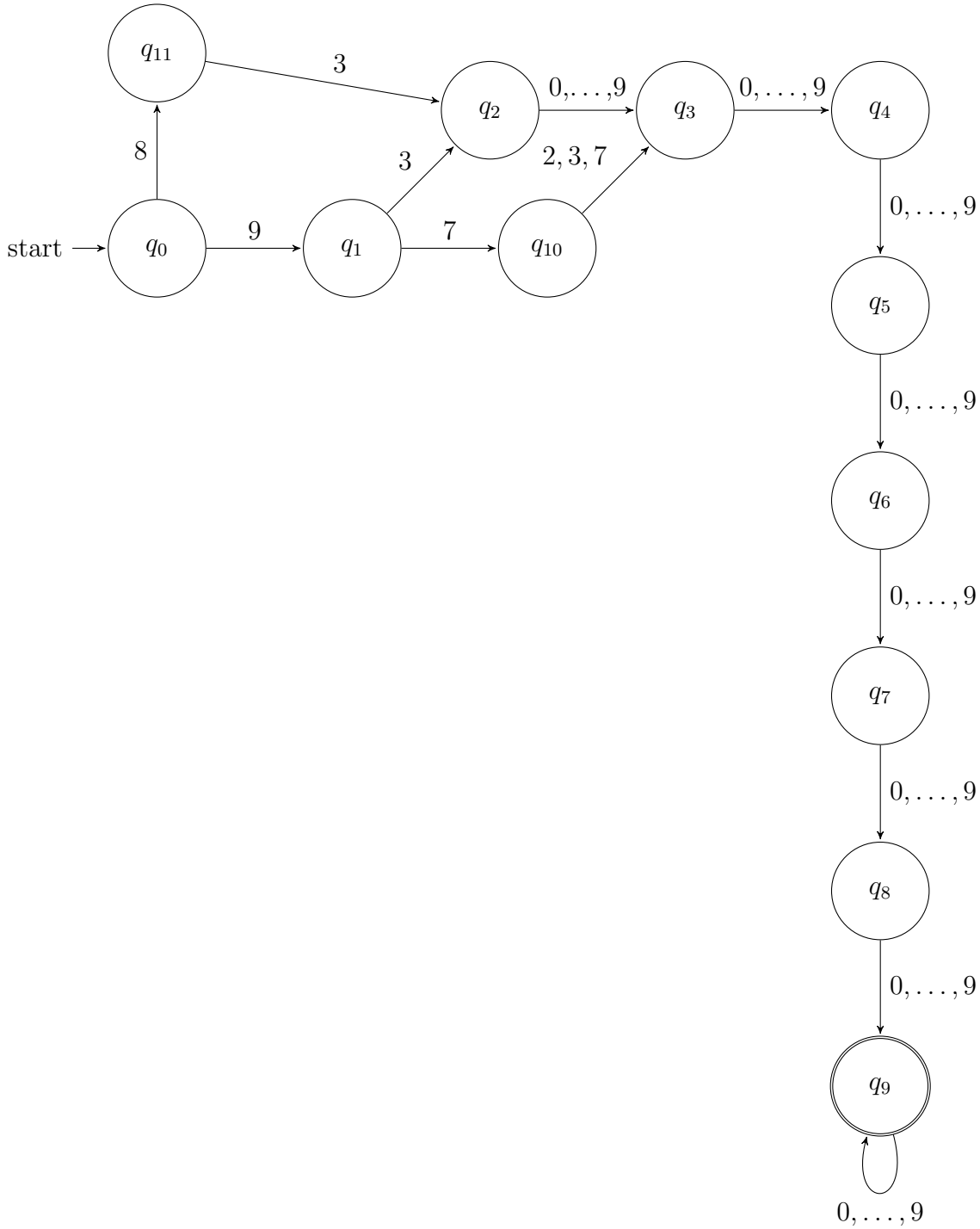
(b) Definimos el siguiente autómata indeterminista, que consta de los estados q_0, q_1, q_2, q_3, q_4 y q_5, donde q_0 es el estado inicial y q_2 y q_5 son los estados aceptadores.



Problema 4. (a) Definir un autómata indeterminista que reconozca los números de teléfono de las provincias de Catalunya.

(b) Explicar cómo, utilizando el autómata del apartado (a), se puede escribir un programa en JAVA para reconocer los números de teléfono de Catalunya.

Solución: (a) El siguiente autómata indeterminista reconoce los números de teléfono de las provincias de Catalunya. Recordemos que el prefijo para llamar a Girona es el 972, a Lleida es el 973 y a Tarragona es el 977.


q_{11}
3
q_2
0,. . . ,9
q_3
0, . . . , 9
q_4
8
3
2, 3, 7
0, . . . , 9
start
q_0
9
q_1
7
q_{10}
q_5
0, . . . , 9
q_6
0, . . . , 9
q_7
0, . . . , 9
q_8
0, . . . , 9
q_9
0, . . . , 9

(b) En primer lugar, hemos de eliminar el indeterminismo del autómata del apartado (a). Para ello, podemos añadir un estado de error, al cual llegaremos desde cualquier otro estado cuando entre un símbolo distinto a los indicados en las transiciones del grafo del autómata del apartado (a). Entonces, el programa asociado a este autómata determinista es el programa buscado para reconocer los números de teléfono de las provincias de Catalunya

Problema 5. Consideremos el autómata indeterminista $M = (\{P, Q, R, S\}, \{a, b\}, \Delta, P, \{P, Q\})$ donde Δ está definida por la siguiente tabla:

P	a	S
P	a	Q
P	λ	Q
Q	b	Q
Q	λ	R
R	b	P
S	a	S
S	b	R

Siguiendo el método visto en clase, trasformar el autómata M en un autómata determinista equivalente.

Solución:

Calculamos primero los cierres de los estados. Se tiene que $\Lambda(P) = PQR$, $\Lambda(Q) = QR$, $\Lambda(R) = R$ y $\Lambda(S) = S$. Construimos entonces el autómata determinista M' equivalente a M. El estado inicial de M' es $\Lambda(P) = PQR$. Construimos la función de transición δ' para M'.

$\delta'(PQR, a) = \Lambda(Q) \cup \Lambda(S) = QRS$,
$\delta'(PQR, b) = \Lambda(P) \cup \Lambda(Q) = PQR$,
$\delta'(QRS, a) = \Lambda(S) = S$,
$\delta'(QRS, b) = \Lambda(P) \cup \Lambda(Q) \cup \Lambda(R) = PQR$,
$\delta'(S, a) = \Lambda(S) = S$,
$\delta'(S, b) = \Lambda(R) = R$,
$\delta'(R, a) = \emptyset$,
$\delta'(R, b) = \Lambda(P) = PQR$,
$\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$.

Por tanto, los estados de M' son: PQR, QRS, S, R y $\emptyset$. Como P y Q son los estados aceptadores de M, los estados aceptadores de M' son PQR y QRS.

Problema 6. Siguiendo el método visto en clase, construir un autómata determinista equivalente al autómata indeterminista $M = (\{P, Q, R\}, \{a, b, c\}, \Delta, P, \{R\})$ donde Δ está definida por la siguiente tabla:

P	a	P
P	b	Q
P	c	R
Q	λ	P
Q	a	Q
Q	b	R
R	λ	Q
R	a	R
R	c	P

Solución: Calculamos primero los cierres de los estados. Se tiene que $\Lambda(P) = P$, $\Lambda(Q) = PQ$, y $\Lambda(R) = PQR$. Construimos entonces el autómata determinista M' equivalente a M. El estado inicial de M' es $\Lambda(P) = P$. Construimos la función de transición δ' para M'.

$\delta'(P, a) = \Lambda(P) = P$, $\delta'(P, b) = \Lambda(Q) = PQ$, $\delta'(P, c) = \Lambda(R) = PQR$,

$\delta'(PQ, a) = \Lambda(P) \cup \Lambda(Q) = PQ$, $\delta'(PQ, b) = \Lambda(Q) \cup \Lambda(R) = PQR$, $\delta'(PQ, c) = \Lambda(R) = PQR$,

$\delta'(PQR, a) = \Lambda(P) \cup \Lambda(Q) \cup \Lambda(R) = PQR$, $\delta'(PQR, b) = \Lambda(Q) \cup \Lambda(R) = PQR$, $\delta'(PQR, c) = \Lambda(P) \cup \Lambda(R) = PQR$.

Por tanto, los estados de M' son: P, PQ y PQR. Como R es el único estado aceptador de M, PQR es el único estado aceptador de M'.

Problema 7. Consideremos el autómata indeterminista $M = (\{A, B, C, D, E\}, \{0, 1\}, \Delta, A, \{B, C\})$ donde Δ está definida por la siguiente tabla:

A	λ	B
A	λ	C
B	0	D
B	1	B
C	0	C
C	1	E
D	0	B
D	1	D
E	0	E
E	1	C

Se pide entonces:

(1) Describir el lenguaje $L(M)$.

(2) Siguiendo el método visto en clase, transformar el autómata M en un autómata determinista equivalente.

(3) Programar en JAVA el autómata determinista obtenido en (2).

Solución:

(1) Se observa que el estado B reconoce las palabras de bits que tienen un número par de ceros, y el estado C reconoce las palabras de bits que tienen un número par de unos. Por tanto, como B y C son los estados aceptadores de M, tenemos que

$$L(M) = \{x \in \{0, 1\}^* : n_0(x) \text{ es par o } n_1(x) \text{ es par}\}.$$

(2) Se tiene que $\Lambda(A) = \{A, B, C\} = ABC$, $\Lambda(B) = B$, $\Lambda(C) = C$, $\Lambda(D) = D$ y $\Lambda(E) = E$. Construimos entonces el autómata determinista M' equivalente a M. El estado inicial de M' es $\Lambda(A) = ABC$. Definimos ahora la función de transición δ' para M'.

$\delta'(ABC, 0) = \Lambda(C) \cup \Lambda(D) = CD$,
$\delta'(ABC, 1) = \Lambda(B) \cup \Lambda(E) = BE$,
$\delta'(CD, 0) = \Lambda(B) \cup \Lambda(C) = BC$,

$\delta'(CD, 1) = \Lambda(D) \cup \Lambda(E) = DE$,
$\delta'(BE, 0) = \Lambda(D) \cup \Lambda(E) = DE$,
$\delta'(BE, 1) = \Lambda(B) \cup \Lambda(C) = BC$.
$\delta'(BC, 0) = \Lambda(C) \cup \Lambda(D) = CD$,
$\delta'(BC, 1) = \Lambda(B) \cup \Lambda(E) = BE$,
$\delta'(DE, 0) = \Lambda(B) \cup \Lambda(E) = BE$,
$\delta'(DE, 1) = \Lambda(C) \cup \Lambda(D) = CD$.

Por tanto, los estados de M' son: ABC, CD, BE, BC y DE. Como B y C son los estados aceptadores de M, los estados aceptadores de M' son ABC, CD, BE y BC.

(3) Representamos a ABC por 0, a CD por 1, a BE por 2, a BC por 3 y a DE por 4. Podemos escribir entonces el siguiente programa 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 == '0') q = 1; else if (c == '1') q = 2;
        break;
        case 1:
        if (c == '0') q = 3; else if (c == '1') q = 4;
        break;
        case 2:
        if (c == '0') q = 4; else if (c == '1') q = 3;
        break;
        case 3:
        if (c == '0') q = 1; else if (c == '1') q = 2;
        break;
        case 4:
        if (c == '0') q = 2; else if (c == '1') q = 1;
        break; }
  c = entrada.charAt(++i); }
  if (q == 4) return false; else return true; }
```

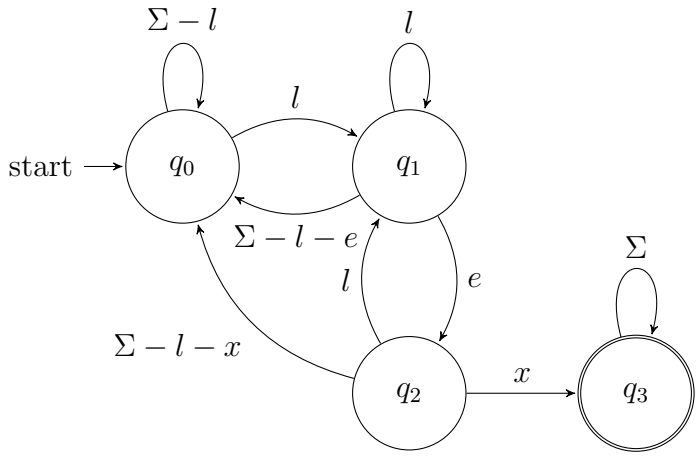
Problema 8. (a) Definir un autómata determinista que determine si un texto de caracteres contiene el patrón "lex".

(b) Definir un autómata indeterminista que determine si un texto de caracteres contiene el patrón "lex" o el patrón "ens2001".

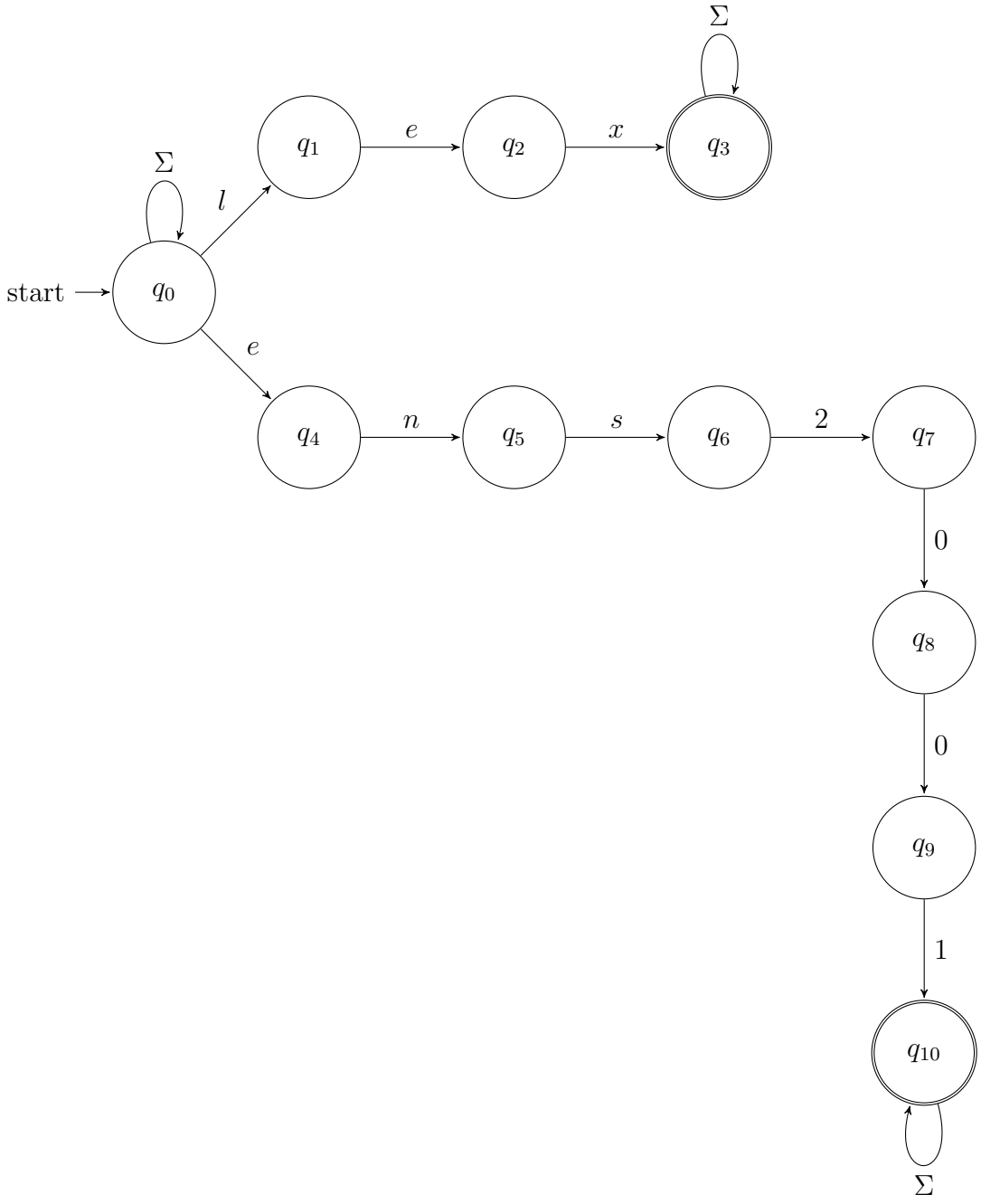
(c) Programar en Java el autómata determinista definido en (a).

Solución:

(a) Denotamos por Σ al conjunto de los caracteres ASCII. El siguiente autómata determinista determina entonces si un texto de caracteres contiene el patrón "lex".



(b) Denotamos por Σ al conjunto de los caracteres ASCII. El siguiente autómata indeterminista determina entonces si un texto de caracteres contiene el patrón "lex" o el patrón "ens2001".


start
q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
q_9
q_{10}
Σ
l
e
x
e
n
s
2
0
0
1
Σ

(c) Podemos escribir el siguiente programa en JAVA para simular el autómata determinista del apartado (a).

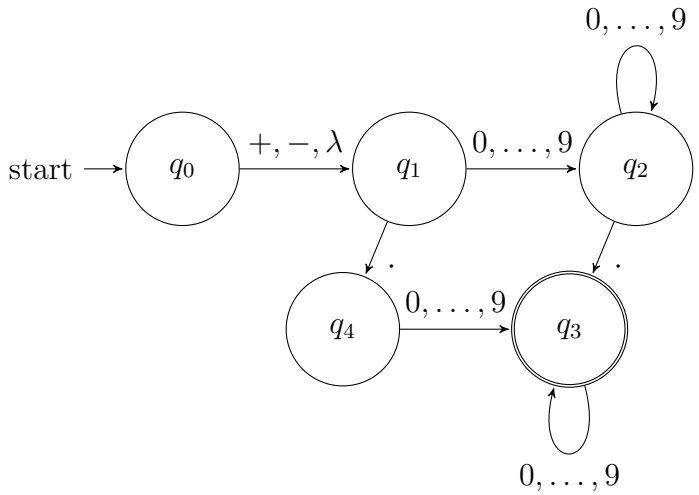
```
public boolean simular (String entrada)
{ int q = 0, i = 0;
  char c = entrada.charAt(0);
  while (c != '$')
  { switch(q)
        { case 0:
        if (c == 'l') q = 1;
        break;
        case 1:
        if (c == 'e') q = 2; else if ((c != 'e') && (c != 'l')) q = 0;
        break;
        case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

Problema 9. (a) Definir un autómata indeterminista para reconocer números decimales que contengan: (a) un signo + o − opcional; (b) una palabra de dígitos; (c) un punto decimal; (d) una segunda palabra de dígitos. Tanto la primera palabra de dígitos como la segunda pueden estar vacías, pero al menos una de las dos palabras no puede estar vacía.

(b) Explicar, cómo utilizando el autómata del apartado (a), se puede escribir un programa en JAVA para reconocer números decimales (el tipo "float").

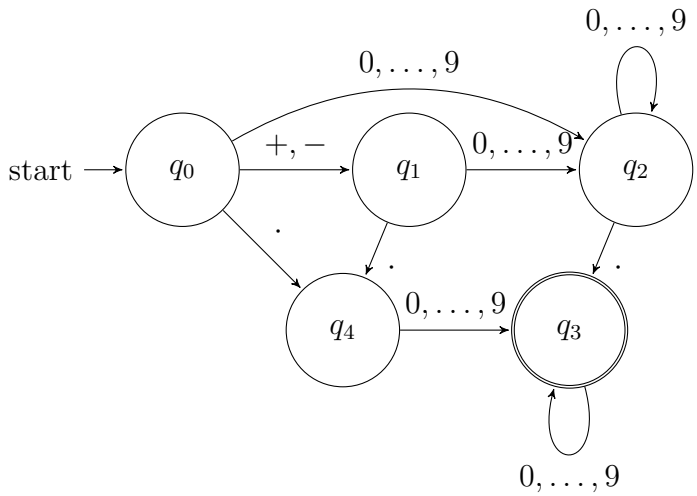
Solución:

(a) En primer lugar, definimos el siguiente autómata indeterminista M_1, en donde q_0 es el estado inicial, y q_3 es el único estado aceptador.



(b) Para escribir un programa en JAVA que reconozca los números decimales, en primer lugar hemos de eliminar el indeterminismo del autómata M_1 del apartado (a). Para ello, podemos utilizar el algoritmo visto en clase de teoría para transformar un autómata indeterminista en un autómata determinista equivalente. Sin embargo, en este caso podemos construir di-

rectamente el autómata determinista equivalente. Para ello, construimos el siguiente autómata M_2 equivalente a M_1, en el que eliminamos la transición con la λ.



Ahora, podemos definir el autómata determinista equivalente, simplemente añadiendo a este último autómata M_2 un estado de error, al cual llegaremos desde cualquier otro estado cuando entre un carácter distinto a los indicados en las transiciones del gráfico del autómata M_2. El programa asociado a este autómata determinista será entonces el programa buscado para reconocer los números decimales.

Problema 10. Explicar cómo diseñar un analizador léxico para reconocer las siguientes categorías sintácticas:

(i) identificadores formados por letras y dígitos de manera que el primer carácter es una letra,

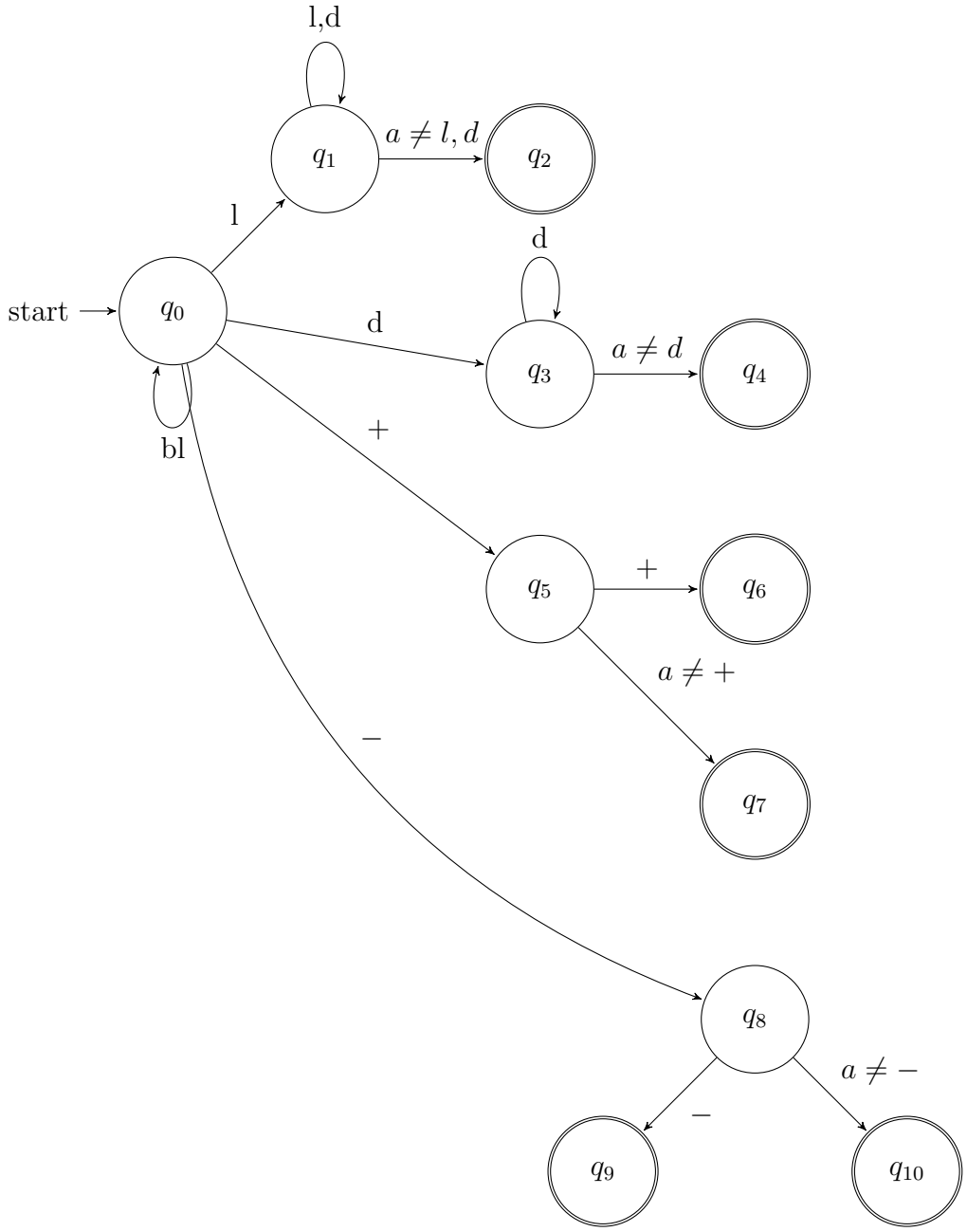
(ii) números enteros sin signo,

(iii) los operadores + y −,

(iv) los operadores ++ y −−.

Solución:

En primer lugar, construimos el siguiente autómata indeterminista M para reconocer las categorías sintácticas indicadas, en el cual los estados aceptadores son q_2, q_4, q_6, q_7, q_9 y q_{10}. Representamos por bl al carácter blanco.



Por tanto, el estado q_2 reconoce un identificador formado por letras y dígitos, el estado q_4 reconoce un entero sin signo, el estado q_6 reconoce $++$, el estado q_7 reconoce $+$, el estado q_9 reconoce $--$ y el estado q_{10} reconoce

el operador $-$. Se observa que los estados q_2, q_4, q_7 y q_{10} van adelantados un carácter, ya que dichos estados reconocen la categoría sintáctica al leer el símbolo siguiente a la categoría. Entonces, para escribir el analizador léxico, en primer lugar hemos de añadir un estado de error al autómata M, al cual llegaremos desde cualquier otro estado cuando entre un carácter distinto a los indicados en las transiciones del gráfico del autómata M. A continuación, programamos el autómata resultante, siguiendo el método visto en clase, con las siguientes modificaciones:

- Cuando se llegue a un estado aceptador, el programa imprimirá la categoría sintáctica reconocida y volverá al estado inicial.
- Cuando se llegue a q_2, a q_4, a q_7 o a q_{10}, el programa no leerá el siguiente símbolo de la entrada. Y sí lo leerá, cuando llegue a cualquier otro estado.