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

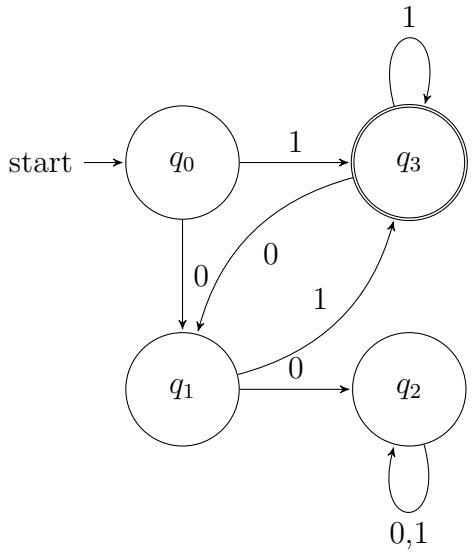
(a) $\{x \in \{0,1\}^* : x$ acaba en 1 y no contiene 00$\}$.

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

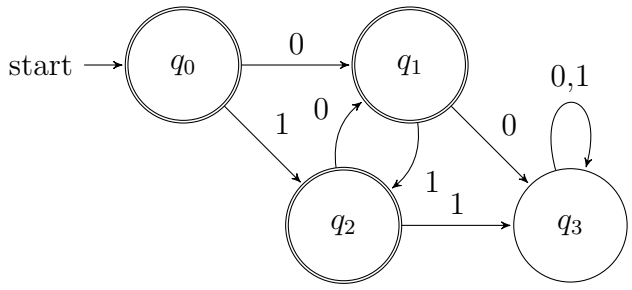
(c) $\{x \in \{0,1\}^* : x$ contiene 01 o 110 $\}$

Solución:

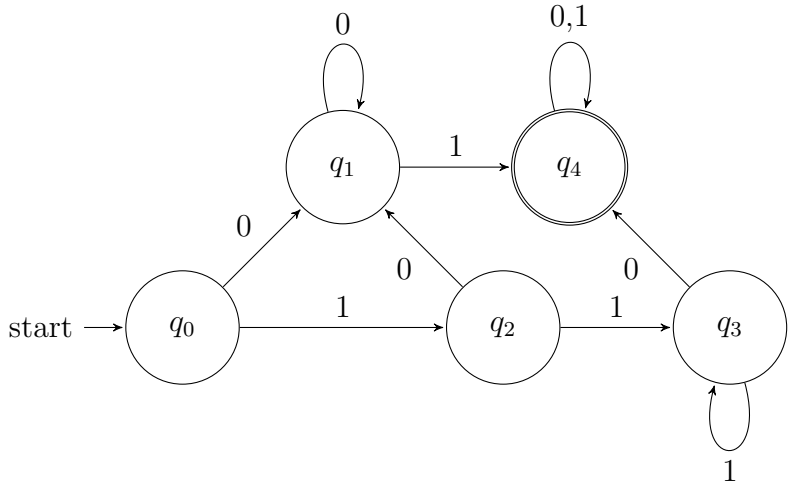
(a) Definimos el siguiente autómata determinista, que consta de los estados q_0, q_1, q_2 y q_3, donde q_0 es el estado inicial y q_3 es el único estado aceptador.



(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.

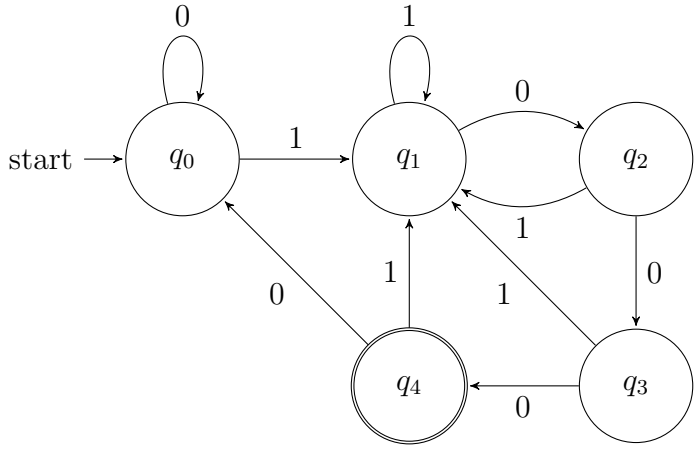


Problema 2. (a) Definir un autómata determinista que reconozca el lenguaje de las palabras de bits que acaban en 1000.

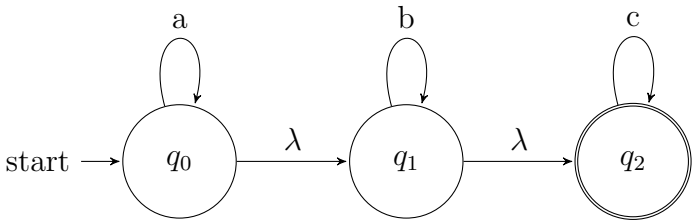
(b) Definir un autómata indeterminista que reconozca el lenguaje de las palabras con cero o más letras a, seguidas de cero o más letras b, seguidas de cero o más letras c .

Solución:

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



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

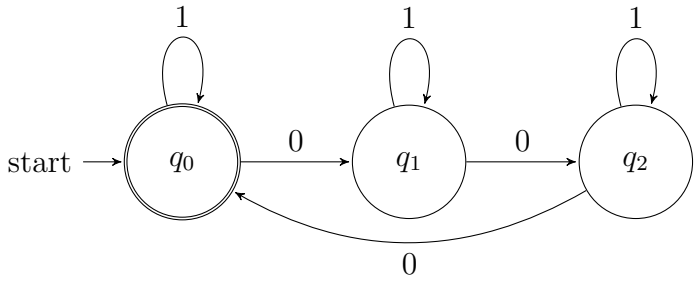

a
b
c
start
q_0
λ
q_1
λ
q_2

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

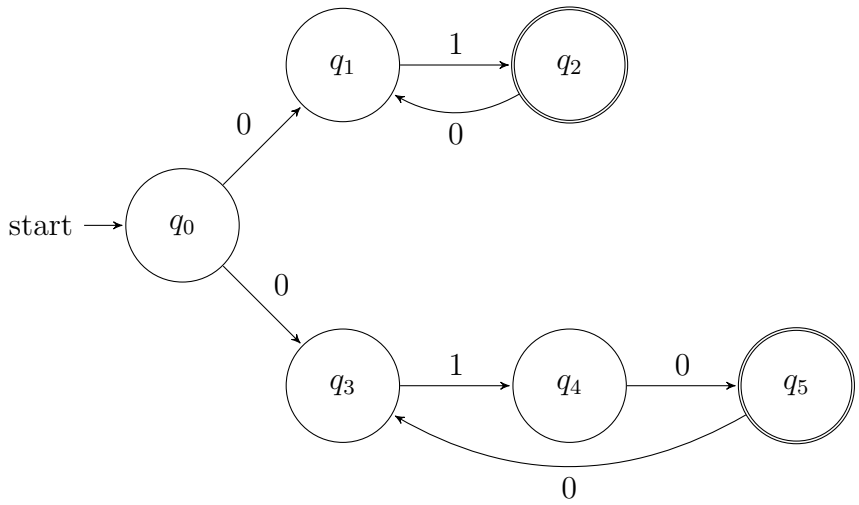
(b) Definir un autómata indeterminista M tal que $L(M) = \{(01)^n : n \geq 1\} \cup \{(010)^n : n \geq 1\}$, es decir, $L(M)$ es el lenguaje de las palabras formadas por 01 repetido una o más veces, o por 010 repetido una o más veces.

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.


start
q_0
q_1
q_2
q_3
q_4
q_5
0
1
0
0
1
0
0

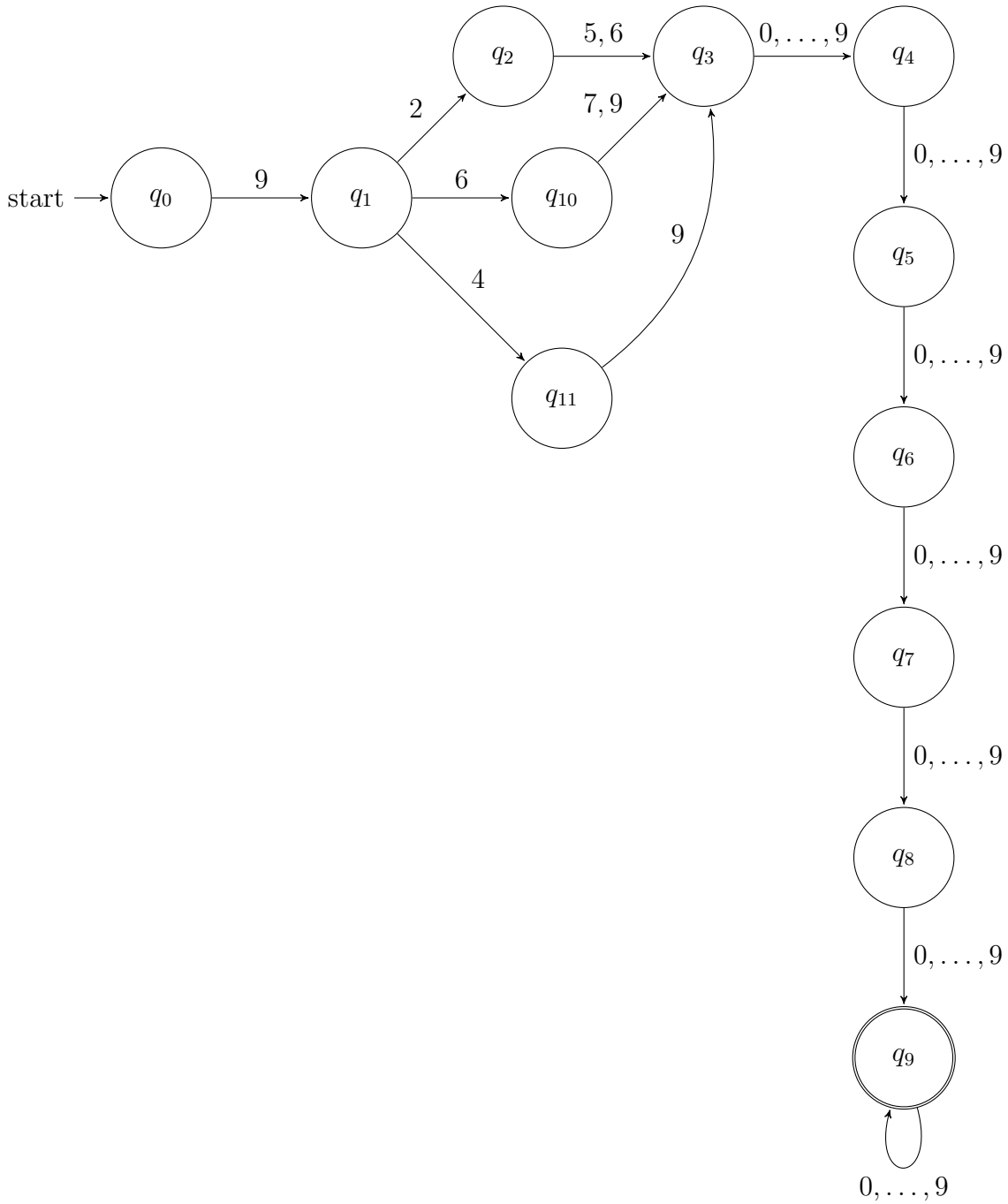
Problema 4. Para llamar por teléfono a una provincia de la comunidad de Castilla-La Mancha hay que marcar un número de seis dígitos y previamente el siguiente prefijo: 967 si se llama a Albacete, 969 si se llama a Cuenca, 949 si se llama a Guadalajara, 925 si se llama a Toledo, y 926 si se llama a Ciudad Real. Se pide entonces:

(a) Definir un autómata indeterminista que reconozca los números de teléfono de la comunidad de Castilla-La Mancha.

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

Solución:

(a) El siguiente autómata indeterminista, en donde q_9 es el único estado aceptador, reconoce los números de teléfono de Castilla-La Mancha.


start
q_0
9
q_1
2
q_2
5, 6
q_3
0, . . . , 9
q_4
6
q_{10}
7, 9
4
q_{11}
9
0, . . . , 9
q_5
0, . . . , 9
q_6
0, . . . , 9
q_7
0, . . . , 9
q_8
0, . . . , 9
q_9
0, . . . , 9

(b) Para escribir un programa en JAVA que reconozca los números de teléfono de Castilla-La Mancha, en primer lugar hemos de eliminar el indeterminismo del autómata del apartado (a). Podemos hacer esto directamente añadiendo un estado de error al autómata, al cual llegaremos desde cualquier otro estado cuando entre un dígito distinto a los indicados en las transiciones del gráfico del autómata del apartado (a). El programa asociado a este autómata determinista será entonces el programa buscado para reconocer los números de teléfono de Castilla-La Mancha.

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

A	1	B
A	λ	C
B	0	B
B	0	C
B	1	C
C	0	A

Se pide entonces:

(1) Determinar, razonando la respuesta, si las siguientes palabras son reconocidas por M: 1100, 1101, 01010.

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

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

Solución:

(1) 1100 es reconocida por el siguiente cómputo: $A1100 \vdash_M B100 \vdash_M C00 \vdash_M A0 \vdash_M C0 \vdash_M A$.

1101 no es reconocida. Hay dos cómputos posibles. En uno de ellos, se pasa del estado A al estado B leyendo el primer uno, a continuación se pasa del estado B al estado C leyendo el segundo uno, a continuación se pasa del estado C al estado A leyendo el cero, y a continuación se pasa del estado A al estado C sin leer ningún símbolo, quedando entonces el cómputo bloqueado en el estado C. Y en el otro cómputo, se pasa del estado A al estado B leyendo el primer uno, a continuación se pasa del estado B al estado C leyendo el segundo uno, a continuación se pasa del estado C al estado A leyendo el cero, y a continuación se pasa del estado A al estado B leyendo el último uno. Como B no es aceptador, este segundo cómputo tampoco reconoce la palabra.

01010 es reconocida por el siguiente cómputo: $A01010 \vdash_M C01010 \vdash_M A1010 \vdash_M B010 \vdash_M B10 \vdash_M C0 \vdash_M A$.

(2) Se tiene que $\Lambda(A) = AC$, $\Lambda(B) = B$, $\Lambda(C) = C$. Construimos entonces el autómata determinista M' equivalente a M. El estado inicial de

M' es $\Lambda(A) = AC$. Construimos la función de transición δ' para M'.

$\delta'(AC, 0) = \Lambda(A) = AC$,
$\delta'(AC, 1) = \Lambda(B) = B$,
$\delta'(B, 0) = \Lambda(B) \cup \Lambda(C) = BC$,
$\delta'(B, 1) = \Lambda(C) = C$,
$\delta'(BC, 0) = \Lambda(A) \cup \Lambda(B) \cup \Lambda(C) = ABC$,
$\delta'(BC, 1) = \Lambda(C) = C$,
$\delta'(C, 0) = \Lambda(A) = AC$,
$\delta'(C, 1) = \emptyset$,
$\delta'(ABC, 0) = \Lambda(A) \cup \Lambda(B) \cup \Lambda(C) = ABC$,
$\delta'(ABC, 1) = \Lambda(B) \cup \Lambda(C) = BC$,
$\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$.

Por tanto, los estados de M' son: AC, B, BC, C, ABC y $\emptyset$. Como A es el único estado aceptador de M, los estados aceptadores de M' son AC y ABC.

(3) Representamos al estado AC por 0, al estado B por 1, al estado C por 2, al estado BC por 3 y al estado ABC 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 == '1') q = 1;
        break;
        case 1:
        if (c == '0') q = 3; else if (c == '1') q = 2;
        break;
        case 2:
        if (c == '0') q = 0; else if (c == '1') return false;
        break;
        case 3:
        if (c == '0') q = 4; else if (c == '1') q = 2;
        break;
```

```
        case 4:
        if (c == '1') q = 3;
        break;}
c = entrada.charAt(++i); }
if (q == 0 || q == 4) return true; else return false; }
```

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

A	0	A
A	λ	B
B	0	C
B	λ	D
C	1	B
D	0	D

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 o en C el autómata determinista obtenido en (2).

Solución:

(1) Se observa que si estamos en el estado A se habrá leído un bloque de ceros, es decir, una palabra de la forma 0^n para algún $n \geq 0$. Si estamos en el estado B, se habrá leído una palabra de la forma $0^n(01)^m$ para $n, m \geq 0$. Y si estamos en el estado D, se habrá leído una palabra de la forma $0^n(01)^m0^k$ para $n, m, k \geq 0$. Como D es el único estado aceptador del autómata, se tiene que $L(M) = \{0^n(01)^m0^k : n, m, k \geq 0\}$.

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

$\delta'(ABD, 0) = \Lambda(A) \cup \Lambda(C) \cup \Lambda(D) = ABCD$,
$\delta'(ABD, 1) = \emptyset$,
$\delta'(ABCD, 0) = \Lambda(A) \cup \Lambda(C) \cup \Lambda(D) = ABCD$,
$\delta'(ABCD, 1) = \Lambda(B) = BD$,
$\delta'(\emptyset, 0) = \delta'(\emptyset, 1) = \emptyset$,
$\delta'(BD, 0) = \Lambda(C) \cup \Lambda(D) = CD$,
$\delta'(BD, 1) = \emptyset$,

$\delta'(CD, 0) = \Lambda(D) = D$,
$\delta'(CD, 1) = \Lambda(B) = BD$,
$\delta'(D, 0) = \Lambda(D) = D$,
$\delta'(D, 1) = \emptyset$.

Por tanto, los estados de M' son: $ABD, ABCD, BD, CD, D$ y $\emptyset$. Como D es el único estado aceptador de M, los estados aceptadores de M' son $ABD, ABCD, BD, CD$ y D (es decir, todos los estados salvo $\emptyset$).

(3) Representamos al estado ABD por 0, al estado $ABCD$ por 1, al estado BD por 2, al estado CD por 3 y al estado D por 4. Como $\emptyset$ es un estado de error, no hace falta representarlo. 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 return false;
        break;
        case 1:
        if (c == '1') q = 2;
        break;
        case 2:
        if (c == '0') q = 3; else return false;
        break;
        case 3:
        if (c == '0') q = 4; else q = 2;
        break;
        case 4:
        if (c == '1') return false;
        break;}
  c = entrada.charAt(++i); }
  return true; }
```

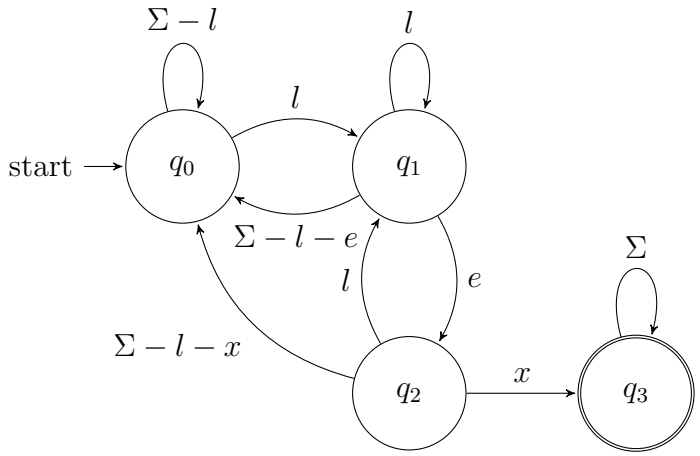
Problema 8. (a) Definir un autómata determinista que determine si un texto de caracteres contiene el patrón "lex".

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

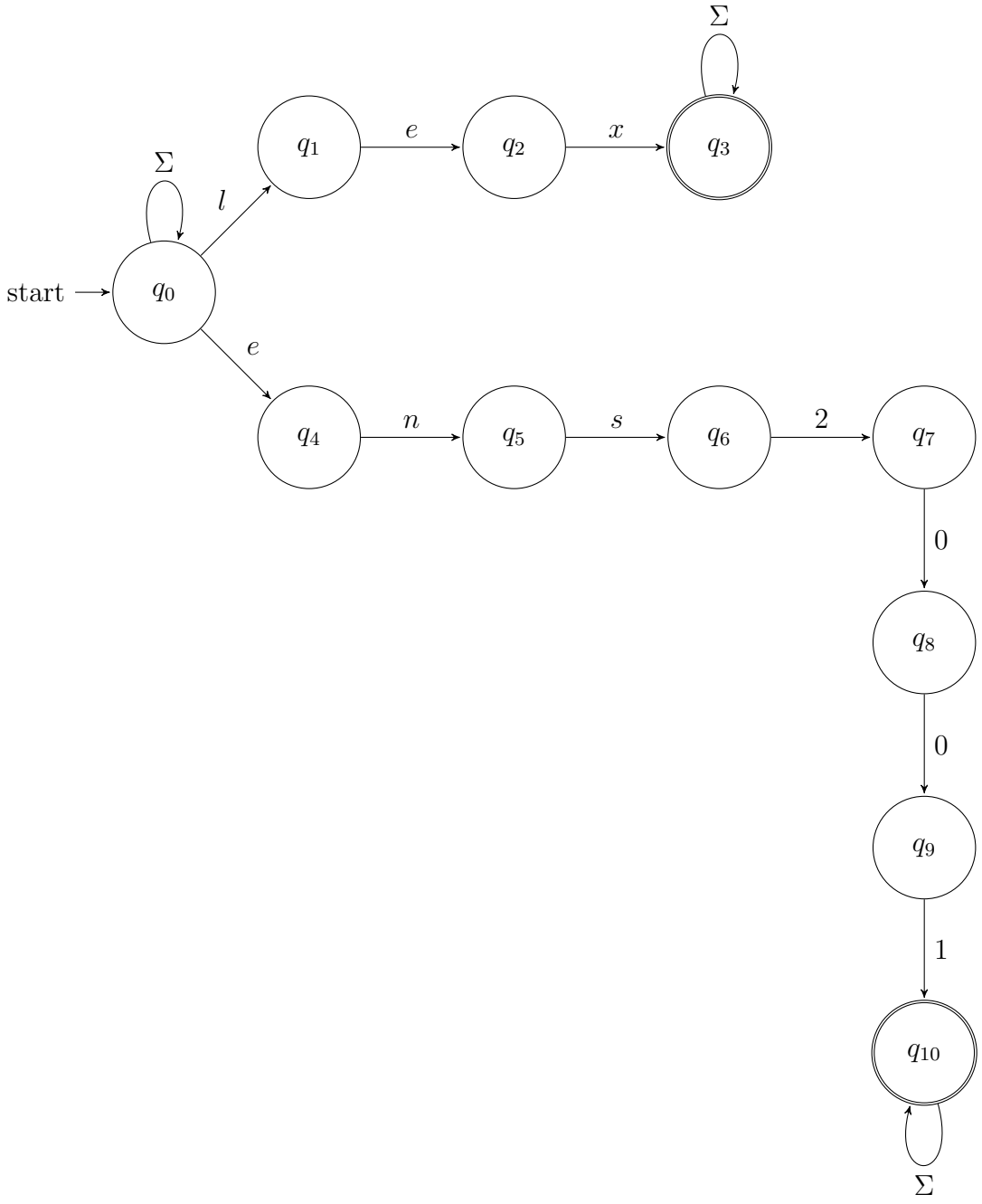
(c) Programar en Java el autómata determinista definido en (a).

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).

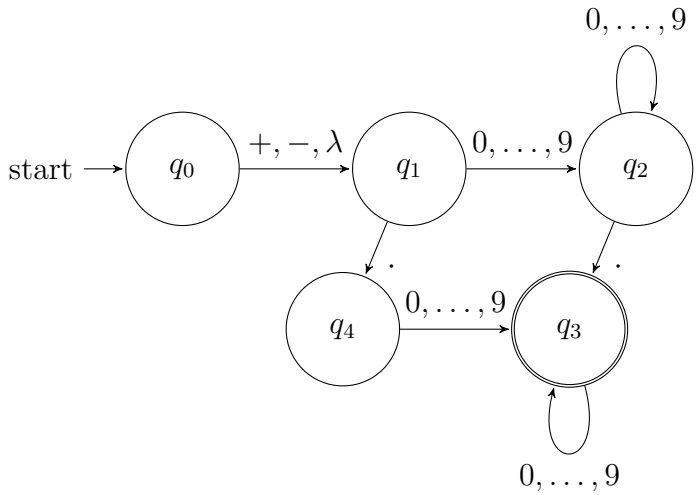
```
public boolean simular (String entrada)
{ int q = 0, i = 0;
  char c = entrada.charAt(0);
  while (c != '$')
  { switch(q)
        { case 0:
        if (c == 'l') q = 1;
        break;
        case 1:
        if (c == 'e') q = 2; else if ((c != 'e') && (c != 'l')) q = 0;
        break;
        case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

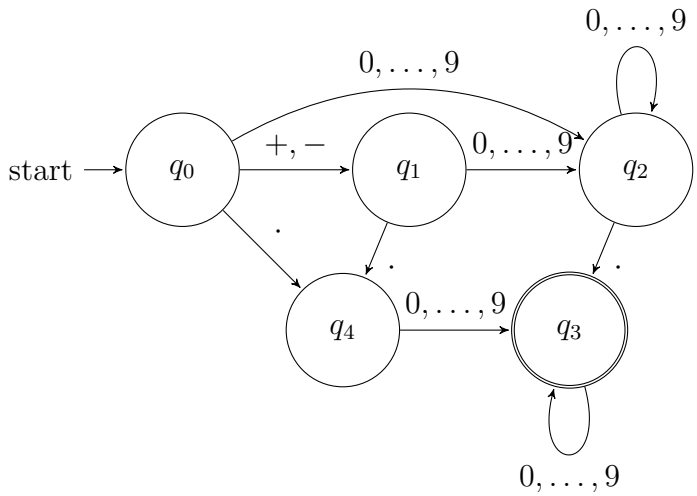
Solución:

(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,

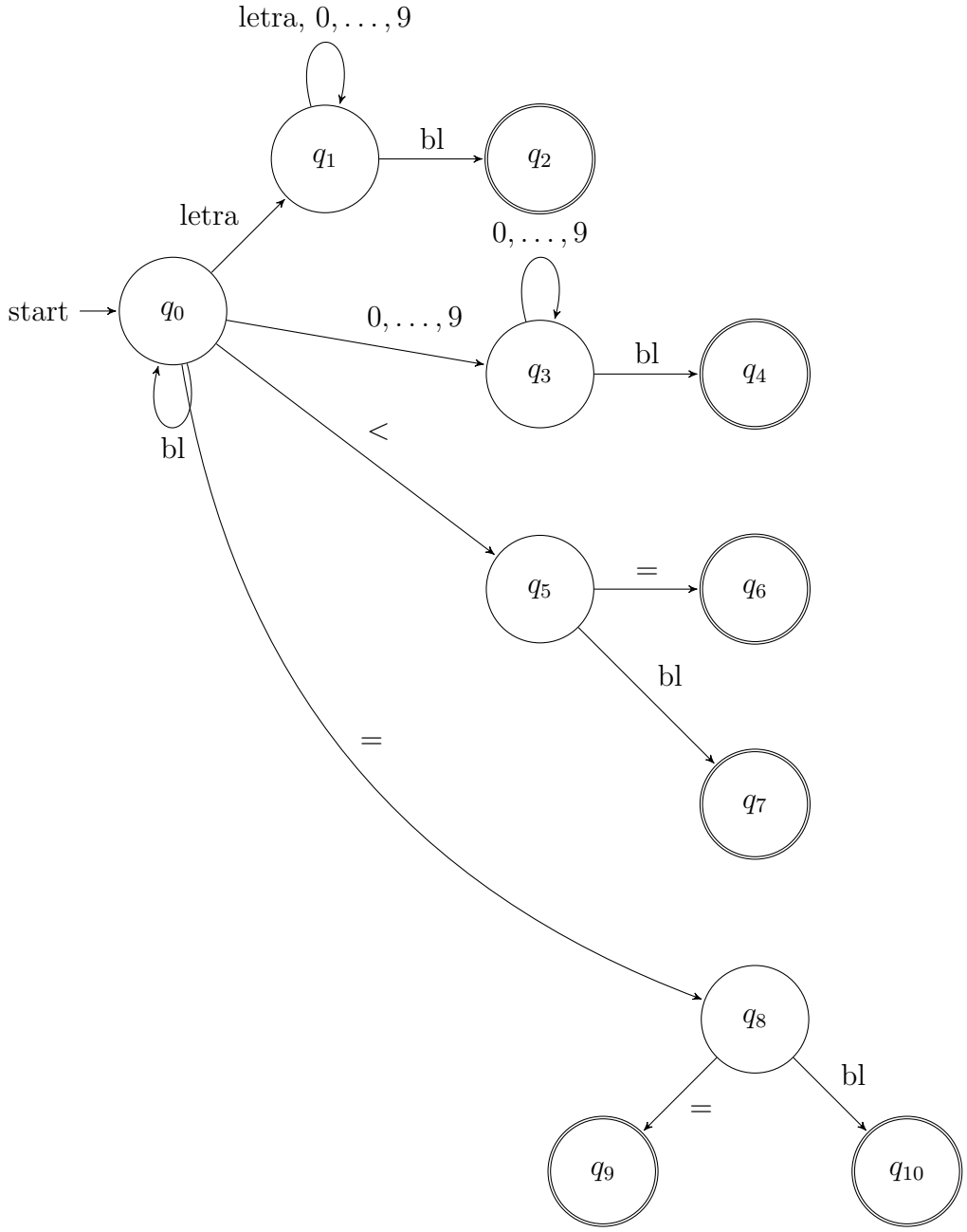
(ii) números enteros sin signo,

(iii) la asignación =,

(iv) los predicados ==, < 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

la asignación =. 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.