Autómatas y Lenguajes Formales

Tema X: Autómatas de divisibilidad

Dr. Favio Ezequiel Miranda Perea

favio@ciencias.unam.mx

Facultad de Ciencias UNAM[1]

12 de enero de 2021

[1] Con el apoyo del proyecto PAPIME PE102117

Autómatas de divisibilidad

Motivación

En el estudio de autómatas finitos deterministas (AFD), una de sus muchas aplicaciones es el verificar si dado un número n es divisible o no por un número k.

Autómatas de divisibilidad

Motivación

En el estudio de autómatas finitos deterministas (AFD), una de sus muchas aplicaciones es el verificar si dado un número n es divisible o no por un número k.

Para mayor comodidad, se usará el número n y k codificado en sistema binario (aunque también se podría hacer cualquier base b). Se construirá un DFA con k estados, donde k es el número por el cual se quiere ver si es divisible o no, en donde cada estado tendrá la función de transición definida para 0 ó 1.

Autómatas de divisibilidad

Motivación

En el estudio de autómatas finitos deterministas (AFD), una de sus muchas aplicaciones es el verificar si dado un número n es divisible o no por un número k.

Para mayor comodidad, se usará el número n y k codificado en sistema binario (aunque también se podría hacer cualquier base b). Se construirá un DFA con k estados, donde k es el número por el cual se quiere ver si es divisible o no, en donde cada estado tendrá la función de transición definida para 0 ó 1.

Asumamos que cada número es "leído" de izquierda a derecha, es decir, el bit más significativo ("msb" *most significative bit* por sus siglas en inglés) es leído primero.

Autómatas de divisibilidad

Motivación

Sea x sea el valor del número binario en algún punto procesando la cadena y sea su residuo r módulo d. Esto significa que x puede ser expresado como combinación lineal como $x = md + r$ para algún entero m. El residuo puede tener valores $0 \leq r < d - 1$. Cada unos de estos residuos corresponden a los estados del autómata.

Autómatas de divisibilidad

Motivación

Sea x sea el valor del número binario en algún punto procesando la cadena y sea su residuo r módulo d. Esto significa que x puede ser expresado como combinación lineal como $x = md + r$ para algún entero m. El residuo puede tener valores $0 \leq r < d - 1$. Cada unos de estos residuos corresponden a los estados del autómata.

Se restringirá el dominio a solo aceptar cadenas no vacías para mayor practicidad.

Autómatas de divisibilidad

Motivación

Para calcular la función de transición verificamos lo siguiente; si se está en un estado q_i, se ha leído q_i (donde q_i es un número en decimal). Si se lee 0, el nuevo número a leer será $2 * q_i$. Si se lee 1, el nuevo número se convierte en $2 * q_i + 1$. El nuevo estado puede ser obtenido restando k (en caso de ser mayor que q_i) de esos valores ($2 * q_i$ ó $2 * q_i + 1$) donde $0 \leq q_i < k$.

Autómatas de divisibilidad

Motivación

Para calcular la función de transición verificamos lo siguiente; si se está en un estado q_i, se ha leído q_i (donde q_i es un número en decimal). Si se lee 0, el nuevo número a leer será $2 * q_i$. Si se lee 1, el nuevo número se convierte en $2 * q_i + 1$. El nuevo estado puede ser obtenido restando k (en caso de ser mayor que q_i) de esos valores ($2 * q_i$ ó $2 * q_i + 1$) donde $0 \leq q_i < k$.

El estado inicial será q_0 y para verficar que si el número ees divisible su estado final (y el único) debe ser q_0 porque significa que el número ya procesado su residuo fue 0.

Autómatas de divisibilidad

Definición formal

Tal autómata (AFD) queda formalmente definido como:

- $Q = \{q_i \mid 0 \leq i < k\}$

Autómatas de divisibilidad

Definición formal

Tal autómata (AFD) queda formalmente definido como:

- $Q = \{q_i \mid 0 \leq i < k\}$
- $\Sigma = \{0, 1\}$

Autómatas de divisibilidad

Definición formal

Tal autómata (AFD) queda formalmente definido como:

- $Q = \{q_i \mid 0 \leq i < k\}$
- $\Sigma = \{0, 1\}$
- $\delta(q_i, 0) = q_{2i \bmod k}$ y $\delta(q_i, 0) = q_{(2i+1) \bmod k}$

Autómatas de divisibilidad

Definición formal

Tal autómata (AFD) queda formalmente definido como:

- $Q = \{q_i \mid 0 \leq i < k\}$
- $\Sigma = \{0, 1\}$
- $\delta(q_i, 0) = q_{2i \mod k}$ y $\delta(q_i, 0) = q_{(2i+1) \mod k}$
- $F = \{q_0\}$

Autómatas de divisibilidad

Definición formal

Tal autómata (AFD) queda formalmente definido como:

- $Q = \{q_i \mid 0 \leq i < k\}$
- $\Sigma = \{0, 1\}$
- $\delta(q_i, 0) = q_{2i \mod k}$ y $\delta(q_i, 0) = q_{(2i+1) \mod k}$
- $F = \{q_0\}$

Autómatas de divisibilidad

Definición formal

Tal autómata (AFD) queda formalmente definido como:

- $Q = \{q_i \mid 0 \leq i < k\}$
- $\Sigma = \{0, 1\}$
- $\delta(q_i, 0) = q_{2i \mod k}$ y $\delta(q_i, 0) = q_{(2i+1) \mod k}$
- $F = \{q_0\}$

Veamos a través de un ejemplo su construcción y después la verificación de un caso en específico:

Ejemplo

Verificar si dado un número n es divisible por 3 o no.

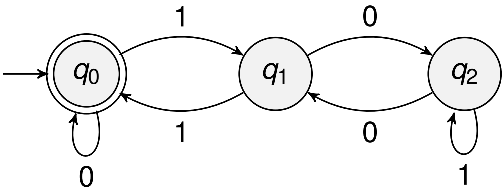
Cualquier número puede escrito de la forma $n = 3 * x + y$ donde x es el cociente y y el residuo. Para 3, hay tres estado en el AFD, cada uno correspondiendo al residuo 0, 1 ó 2. Y para cada estado dos transiciones correpondientes a 0 y 1. La función de transición $\delta(q_i, a)$ nos dice que leyendo del alfabeto a, nos movemos del estado q_i a $q_{i'}$. El estado inicial siempre será q_0, y el estado final indica en residuo, si el estado final es q_0, el número es divisible.

Cabe notar que el último estado es el residuo.

Ejemplo

Verificar si dado un número *n* es divisible por 3 o no.

A continuación se muestra el autómata construido,



Verificar si dado un número n es divisible por 3 o no.

Analicemos por casos

- Se está en el estado q_0 y se lee `0`, nos quedamos en el estado q_0.

Verificar si dado un número *n* es divisible por 3 o no.

Analicemos por casos

- Se está en el estado q_0 y se lee `0`, nos quedamos en el estado q_0.
- Se está en el estado q_0 y se lee `1`, nos movemos al estado q_1 porque el número formado que es `1` y en decimal da de residuo 1.

Verificar si dado un número n es divisible por 3 o no.

Analicemos por casos

- Se está en el estado q_0 y se lee `0`, nos quedamos en el estado q_0.
- Se está en el estado q_0 y se lee `1`, nos movemos al estado q_1 porque el número formado que es `1` y en decimal da de residuo 1.
- Se está en el estado q_1 y se lee `0`, nos movemos al estado q_2 porque el número formado que es `10` y en decimal da de residuo 2.

Verificar si dado un número *n* es divisible por 3 o no.

Analicemos por casos

- Se está en el estado q_0 y se lee `0`, nos quedamos en el estado q_0.
- Se está en el estado q_0 y se lee `1`, nos movemos al estado q_1 porque el número formado que es `1` y en decimal da de residuo 1.
- Se está en el estado q_1 y se lee `0`, nos movemos al estado q_2 porque el número formado que es `10` y en decimal da de residuo 2.
- Se está en el estado q_1 y se lee `1`, nos movemos al estado q_0 porque el número formado que es `11` y en decimal da de residuo 0.

Verificar si dado un número *n* es divisible por 3 o no.

Analicemos por casos

- Se está en el estado q_0 y se lee `0`, nos quedamos en el estado q_0.
- Se está en el estado q_0 y se lee `1`, nos movemos al estado q_1 porque el número formado que es `1` y en decimal da de residuo 1.
- Se está en el estado q_1 y se lee `0`, nos movemos al estado q_2 porque el número formado que es `10` y en decimal da de residuo 2.
- Se está en el estado q_1 y se lee `1`, nos movemos al estado q_0 porque el número formado que es `11` y en decimal da de residuo 0.
- Se está en el estado q_2 y se lee `0`, nos movemos al estado q_1 porque el número formado que es `100` y en decimal da de residuo 1.

Verificar si dado un número n es divisible por 3 o no.

Analicemos por casos

- Se está en el estado q_0 y se lee `0`, nos quedamos en el estado q_0.
- Se está en el estado q_0 y se lee `1`, nos movemos al estado q_1 porque el número formado que es `1` y en decimal da de residuo 1.
- Se está en el estado q_1 y se lee `0`, nos movemos al estado q_2 porque el número formado que es `10` y en decimal da de residuo 2.
- Se está en el estado q_1 y se lee `1`, nos movemos al estado q_0 porque el número formado que es `11` y en decimal da de residuo 0.
- Se está en el estado q_2 y se lee `0`, nos movemos al estado q_1 porque el número formado que es `100` y en decimal da de residuo 1.
- Se está en el estado q_2 y se lee `1`, nos quedamos al estado q_2 porque el número formado que es `101` y en decimal da de residuo 2.

Ejemplo

Verificar si dado un número n es divisible por 3 o no.

Quedando la función de transición como:

δ	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Ejemplo

Verificar si dado un número n es divisible por 3 o no.

Verifiquemos si 4 es divisible por 3: La representación binaria de 6 es `100`, empezando en q_0 se sigue que:

- Estado: q_0, se lee `1`, nuevo estado q_1.

Ejemplo

Verificar si dado un número n es divisible por 3 o no.

Verifiquemos si 4 es divisible por 3: La representación binaria de 6 es `100`, empezando en q_0 se sigue que:

- Estado: q_0, se lee `1`, nuevo estado q_1.
- Estado: q_1, se lee `0`, nuevo estado q_2.

Ejemplo

Verificar si dado un número n es divisible por 3 o no.

Verifiquemos si 4 es divisible por 3: La representación binaria de 6 es `100`, empezando en q_0 se sigue que:

- Estado: q_0, se lee `1`, nuevo estado q_1.
- Estado: q_1, se lee `0`, nuevo estado q_2.
- Estado: q_2, se lee `0`, nuevo estado q_1.

Ejemplo

Verificar si dado un número n es divisible por 3 o no.

Verifiquemos si 4 es divisible por 3: La representación binaria de 6 es `100`, empezando en q_0 se sigue que:

- Estado: q_0, se lee `1`, nuevo estado q_1.
- Estado: q_1, se lee `0`, nuevo estado q_2.
- Estado: q_2, se lee `0`, nuevo estado q_1.

Ejemplo

Verificar si dado un número *n* es divisible por 3 o no.

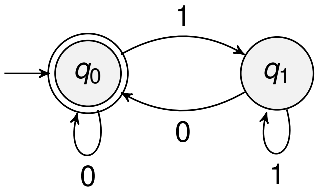
Verifiquemos si 4 es divisible por 3: La representación binaria de 6 es `100`, empezando en q_0 se sigue que:

- Estado: q_0, se lee `1`, nuevo estado q_1.
- Estado: q_1, se lee `0`, nuevo estado q_2.
- Estado: q_2, se lee `0`, nuevo estado q_1.

Como el último estado es q_1, por tanto el número 4 tiene de residuo 1 y no es divisible por 3.

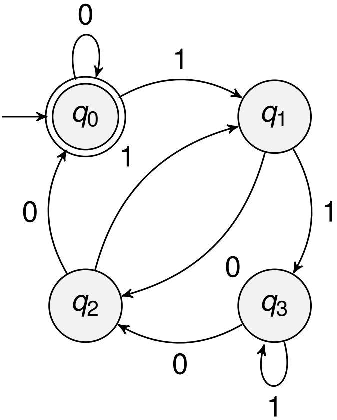
AFD para verificar si un número es divisible por 2

Ejemplo



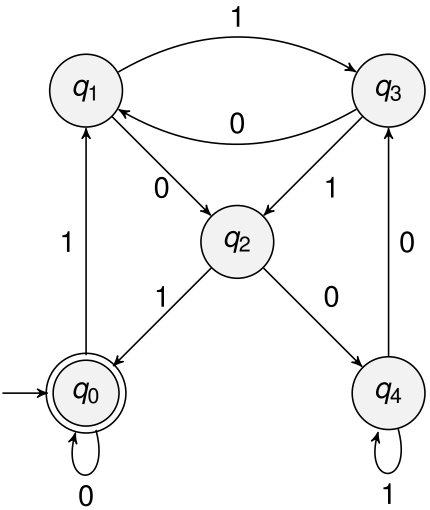
AFD para verificar si un número es divisible por 4

Ejemplo



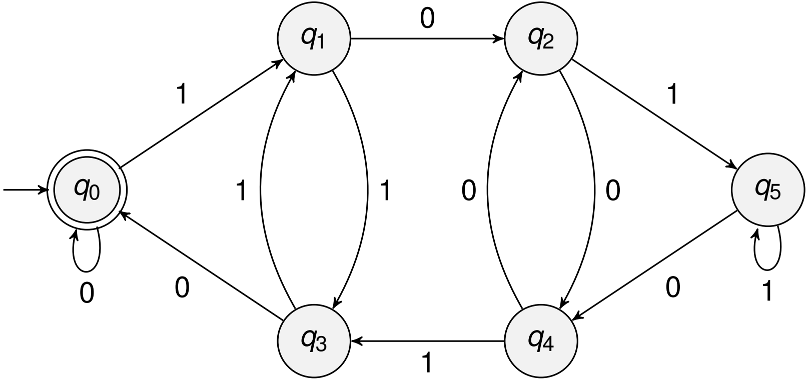
AFD para verificar si un número es divisible por 5

Ejemplo



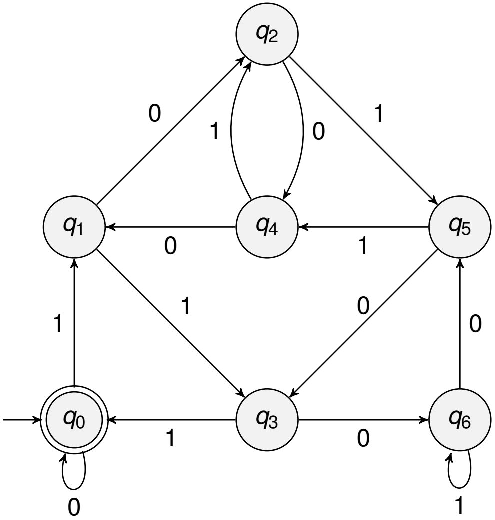
AFD para verificar si un número es divisible por 6

Ejemplo



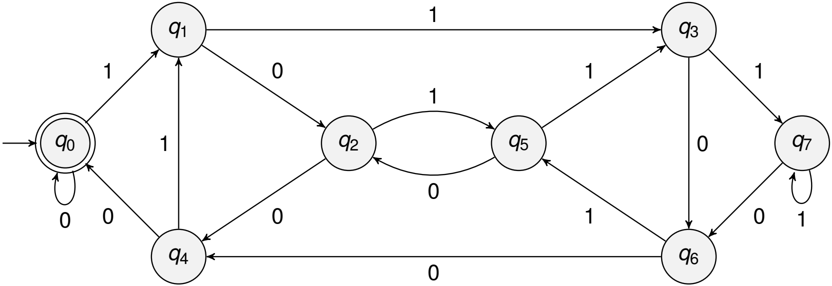
AFD para verificar si un número es divisible por 7

Ejemplo



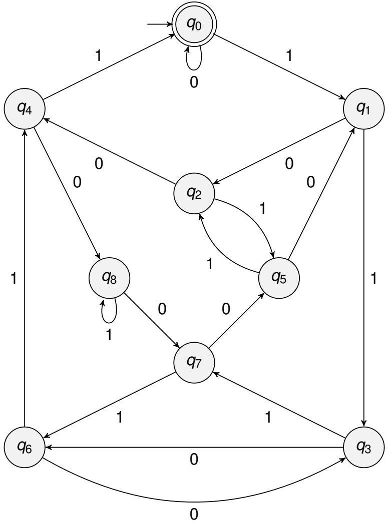
AFD para verificar si un número es divisible por 8

Ejemplo



AFD para verificar si un número es divisible por 9

Ejemplo



Generación del autómata

La pregunta aquí es, ¿se podría hace un programa que construya un autómata que verifique si un número n es divisible por un número k? Claro que sí.

Generación del autómata

La pregunta aquí es, ¿se podría hace un programa que construya un autómata que verifique si un número n es divisible por un número k? Claro que sí.

Generación del autómata

El siguiente programa es mostrado a continuación:

Generación del autómata

El siguiente programa es mostrado a continuación:
Elprograma fue escrito usando *Python* en su versión `3.8.1`, aunque con que sea cualquier posterior a la versión 3 debe de funcionar. Su ejecución es de la siguiente manera una vez ya guardado el archivo:

```
$ python3 dfa-division.py
```

Una vez ejecutado, se introducira linea por linea cada número.
Primero el número *n* y acto seguido el número *k*. Un ejemplo de dicha ejecución es el siguiente:

```
$ python3 dfa-division.py
10
6
```

Dado como resultado:

```
10 is not divisible by 6 and the remainder is 4
```

```python
class DFA(object):
    def __init__(self, k):
        self.k = k
        self.transition_table = self.__fill_transition_table()
    def __fill_transition_table(self):
        automata = [[0, 0] for _ in range(k)] # Representation of the automata in a matrix.
        for state in range(k):  # Computing each transition for every state.
            zero_trans = state << 1  # Next state for 0
            automata[state][0] = zero_trans \
                if zero_trans < self.k else zero_trans - self.k
            one_trans = (state << 1) + 1  # Next state for 1
            automata[state][1] = one_trans \
                if one_trans < self.k else one_trans - self.k
        return automata
    def is_divisible(self, n):
        state = 0  # Initial state
        # Obtain the mask of the most significative bit.
        mask = 1 << (n.bit_length() - 1)
        # Traverse the number in binary format, starting from the most significate bit
        # and in each step obtain the currentt state till there is no string to read.
        while mask != 0:
            current_bit = 1 if mask & n else 0
            state = self.transition_table[state][current_bit]
            mask >>= 1
        return (state == 0, state)
if __name__ == "__main__":
    n = int(input())
    k = int(input())
    dfa = DFA(k)
    is_divisible, remainder = dfa.is_divisible(n)
    if is_divisible:
        print(f"{n} is divisible by {k}")
    else:
        print(f"{n} is not divisible by {k} and the remainder is {remainder}")
```