Ejercicios sobre máquinas de Turing

Juan Diego Walteros Cortes

Programa de ingeniería de sistemas y computación, Universidad de Cundinamarca

Lenguajes y Autómatas 801 ISC TRAN

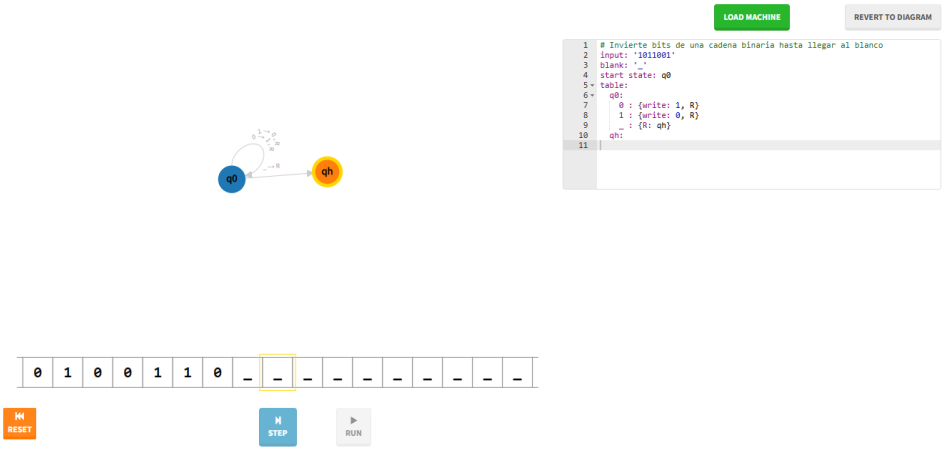
Docente

Ing. Fabio Alejandro Sastoque Rincón

Miércoles 1 de octubre de 2025

Problema 1

Máquina de Turing que proporciona el complemento a 1 de un número binario.



La máquina de Turing implementada tiene como objetivo invertir los bits de un número binario recorriendo la cinta de izquierda a derecha; en cada posición reemplaza los ceros por unos y los unos por ceros, y continúa este proceso hasta encontrar el símbolo blanco que indica el final de la palabra, momento en el cual pasa al estado de aceptación y se detiene, generando como resultado la inversión completa del número binario ingresado.

Problema 2

Diseñar una máquina de Turing que calcula el número consecutivo de un número dado en binario.

Problema2.py > entrada

```python
def maquina_turing_sucesor(binario):
    cinta = list(binario + "_")
    cabeza = len(binario) - 1
    estado = "q0"

    while estado != "qh":
        simbolo = cinta[cabeza]

        if estado == "q0":
            if simbolo == "0":
                cinta[cabeza] = "1"
                estado = "qh"
            elif simbolo == "1":
                cinta[cabeza] = "0"
                cabeza -= 1
            elif simbolo == "_":
                cinta[cabeza] = "1"
                estado = "qh"

    return "".join(cinta).replace("_", "")

entrada = "1011"
salida = maquina_turing_sucesor(entrada)
print("Entrada :", entrada)
print("Salida  :", salida)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
ning/Problema2.py"
Entrada : 1011
Salida  : 1100
```

LOAD MACHINE REVERT TO DIAGRAM

```
input: '1011'        # prueba '1011', '011' ...
blank: '_'
start state: right
table:
  right:
    1: R
    0: R
    _: {L: carry}
  carry:
    1: {write: 0, L}
    0: {write: 1, L: done}
    _: {write: 1, L: done}
  done:
```

_	_	_	_	_	_	_	_	1	1	0	0	_	_	_	_	_

RESET STEP RUN

La máquina de Turing construida tiene como objetivo calcular el sucesor binario de un número, es decir, sumarle uno. Para ello comienza desde el último dígito de la cadena y aplica las reglas del acarreo: si encuentra un 0, lo convierte en 1 y finaliza; si encuentra un 1, lo transforma en 0 y se mueve hacia la izquierda para continuar el acarreo; y si llega al símbolo blanco _, significa que todos los dígitos eran 1, por lo que escribe un 1 al inicio y se detiene. De esta forma, la máquina garantiza el cálculo correcto del número consecutivo en representación binaria.

Problema 3

Diseñar una máquina de Turing que acepta el lenguaje

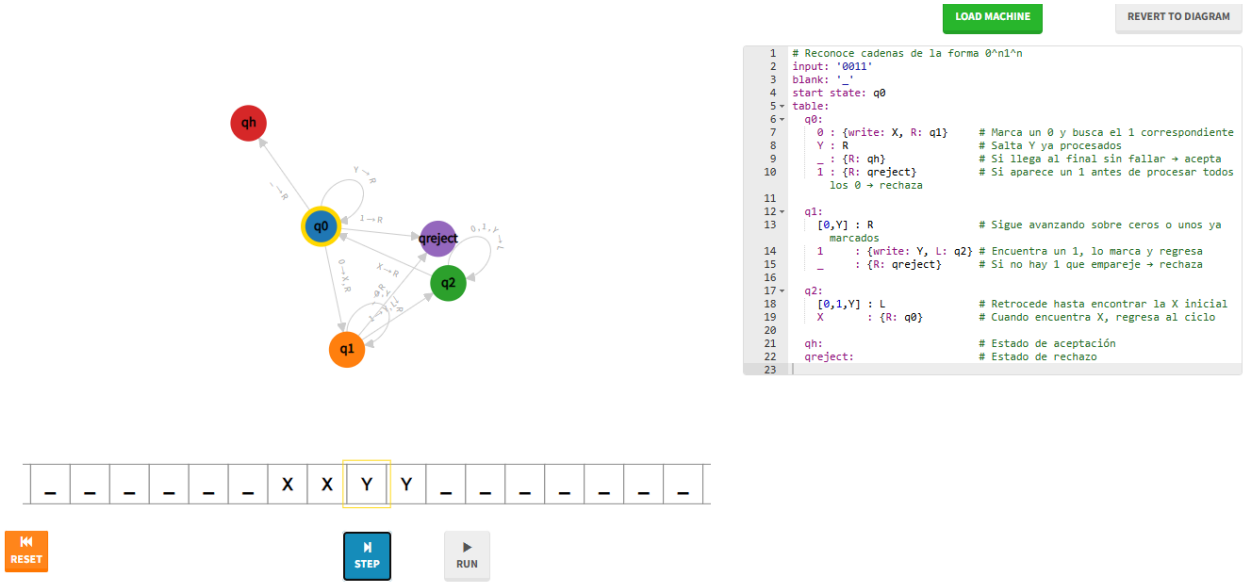
$$L = \{0^n 1^n : n > 0\}$$

Problema3.py > ...

```python
def maquina_turing_0n1n(cadena):
    cinta = list(cadena + "_")
    cabeza = 0
    estado = "q0"

    while True:
        simbolo = cinta[cabeza]

        if estado == "q0":
            if simbolo == "0":
                cinta[cabeza] = "X"
                cabeza += 1
                estado = "q1"
            elif simbolo == "Y":
                cabeza += 1
            elif simbolo == "_":
                return True
            else:
                return False

        elif estado == "q1":
            if simbolo in ["0", "Y"]:
                cabeza += 1
            elif simbolo == "1":
                cinta[cabeza] = "Y"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
0011 => True
000111 => True
011 => False
011 => False
00011 => False
```



La máquina de Turing implementada reconoce cadenas del lenguaje 0n1n0^n1^n0n1n, es decir, aquellas que contienen el mismo número de ceros seguidos por unos. Su funcionamiento consiste en recorrer la cadena, marcar cada 0 inicial con una X, desplazarse a la derecha hasta encontrar el 1 correspondiente, el cual se marca como Y, y luego volver a la izquierda hasta llegar a la X marcada para reiniciar el proceso. Este ciclo se repite hasta que no queden más ceros por procesar; si todos los ceros encuentran su uno correspondiente, la máquina acepta la cadena, mientras que, si aparece un desajuste en el orden o en la cantidad de símbolos, la máquina se detiene en un estado de rechazo.

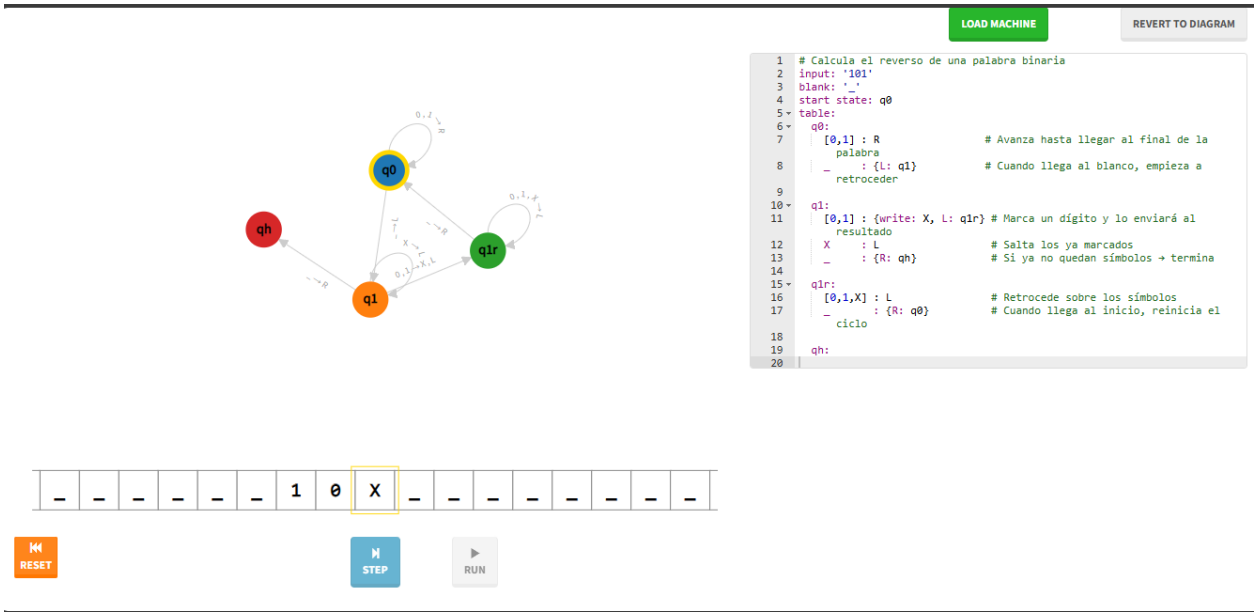
Problema 4

Diseñar una Máquina de Turing que, dada una palabra w del alfabeto $\Sigma=\{0, 1\}$, proporciona su reverso, w_R .

```python
Problema4.py > maquina_turing_reverso
def maquina_turing_reverso(cadena):
    estado = "q0"
    resultado = ""

    while True:
        if estado == "q0":
            if cabeza < len(cinta) and cinta[cabeza] in ["0", "1"]:
                cabeza += 1
            elif cinta[cabeza] == "_":
                cabeza -= 1
                estado = "q1"

        elif estado == "q1":
            if cinta[cabeza] in ["0", "1"]:
                resultado += cinta[cabeza]
                cinta[cabeza] = "X"
                cabeza -= 1
            elif cinta[cabeza] == "X":
                cabeza -= 1
            elif cinta[cabeza] == "_":
                return resultado
ejemplos = ["101", "0110", "111000", "10"]

for e in ejemplos:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
ning/Problema4.py"
101 => 101
0110 => 0110
111000 => 000111
10 => 01
```



La máquina de Turing diseñada tiene como propósito obtener el reverso de una palabra binaria. Para ello, primero se desplaza hacia la derecha hasta encontrar el final de la cadena, y luego empieza a retroceder. Cada vez que encuentra un dígito (0 o 1), lo copia en la salida y lo marca con una X para no procesarlo nuevamente. Después, regresa al inicio de la cinta y repite el

proceso hasta que todos los símbolos han sido marcados, quedando como resultado la palabra original invertida. De esta manera, la máquina garantiza que la salida corresponda exactamente al reverso de la entrada.