|  |
| --- |
| Assignment 1 |
|  |
| Eman Samy  188849 |

# Question 1

|  |
| --- |
| implement a DFA to accept all binary strings that having a substring 00 and ending with 01 This DFA has four states, s the start state and a, b, c. The automation receives an input string and produces an output which is “Accepted” if the string has a substring of “00” and ends with “01”; otherwise, it will output “Rejected”. The accept state is c.  1  0  1 |
| c  b  a  s  1  0  0  1  0 |
| Experimenting with the following input string:  001 Accepted 0101 Rejected  0001 Accepted 00110 Rejected  1001 Accepted  11001 Accepted  0001101 Accepted  1001101 Accepted  100101 Accepted  111001 Accepted Code using python Four functions for the four states s, a, b and c, each state transitions to another state depending on the digit in the input string str. We start when dfa\_state = 0 and loop on the input string until its finished and depending on the state number in the if statements, we go to the next state using the state functions start, state1, state2 and state3 **def start(c):**  **if (c == 0):**  **dfa\_state = 1** **elif (c == 1):**  **dfa\_state = 0** **else:**  **dfa\_state = -1** **return dfa\_state** **def state1( c):** **if (c == 0):** **dfa\_state = 2;** **elif (c == 1):** **dfa\_state = 0** **else:** **dfa\_state = -1;** **return dfa\_state** **def state2(c):** **if (c == 1):** **dfa\_state = 3;** **elif (c == 0):** **dfa\_state = 2;** **else:**  **dfa\_state = -1;** **return dfa\_state** **def state3(c):** **if (c == 0):** **dfa\_state = 2** **elif (c == 1):** **dfa\_state = 1** **else:**  **dfa\_state = -1;** **return dfa\_state** **def DFA(str):** **dfa\_state = 0** **for i in range(len(str)):** **if dfa\_state == 0:** **dfa\_state = start(str[i])** **elif dfa\_state == 1:** **dfa\_state = state1(str[i])** **elif dfa\_state == 2:** **dfa\_state = state2(str[i])** **elif dfa\_state == 3:** **dfa\_state = state3(str[i])** **else:** **return 0** **if dfa\_state == 3:** **return 1** **else:** **return 0** **str = [ 0,0,1,1,0,0,1]****if DFA(str):** **print("Accepted")****else:** print("Rejected")  *Text  Description automatically generatedText  Description automatically generated* |

## Test cases

#1 Input: 001 output: Accepted

Since this substring has consecutive 0’s and ends with 01 at the same time

Graphical user interface, text

Description automatically generated



#2 Input: 11001 output: Accepted

Graphical user interface, text

Description automatically generated



#3 Input: 101001 output: AcceptedGraphical user interface, text, application

Description automatically generated



#4 Input: 110101 output: Rejected

Even though it ends with 01 it is rejected because it doesn’t contain 00

Graphical user interface, text, application

Description automatically generated



#5 Input: 100111 output: Rejected



Even though it contains 00 it is rejected because it doesn’t end with 01

Graphical user interface, text

Description automatically generated



# Question 2

## implement a NFA to accept all strings such that 𝑣𝑎𝑙𝑢𝑒(𝑤) = 𝑣𝑎𝑙𝑢𝑒(𝑤𝑅)

This NFA has three states a, b, c.

We consider the column of a, b and c to be the states while the upper row of a, b and c to be the alphabet on the transition arrow to decide which state are we going to, this NFA can start from anywhere and end anywhere.

a

b

b

a , c

c

b

a

c

b

a

c

c

b

a

Let w = ((axb)xc)xb = (axc)xb = bxb = c

wR = ((bxc)xb)xa = (axb)xa = axa = c Accepted

Let w = ((bxa)xc)xa = (bxc)xb = axa = c Accepted

wR = ((axc)xa)xb = (bxa)xb = cxa = c

Let w = ((cxb)xa)xb = (bxa)xb = bxb = c

wR = ((bxa)xb)xc = (bxb)xc = cxc = c Accepted

Let w = axc = b

wR = cxa = c Rejected

## Code using python

Three functions for the three states a,b,c. Each function outputs a dfa\_state depending on the letter sent to the function. The NFA function loops on the string starting from first letter and depending on the letter, a state is returned as an output from the functions and so on until a final state is reached. Then we reverse the string and send it to the NFA and compare the final state of the normal string and reversed string and if they are equal, “Accepted” message is displayed; otherwise, it is “Rejected”

def stateA(c):

if (c == "a"):

dfa\_state = "c"

elif (c == "b"):

dfa\_state = "a"

elif (c == "c"):

dfa\_state = "b"

else:

dfa\_state = -1

return dfa\_state

def stateB(c):

if (c == "a"):

dfa\_state = "b"

elif (c == "b"):

dfa\_state = "c"

elif (c == "c"):

dfa\_state = "a"

else:

dfa\_state = -1

return dfa\_state

def stateC(c):

if (c == "a"):

dfa\_state = "c"

elif (c == "b"):

dfa\_state = "b"

elif (c == "c"):

dfa\_state = "c"

else:

dfa\_state = -1

return dfa\_state

def NFA(str):

dfa\_state = str[0]

for i in range(len(str)-1):

if dfa\_state == "a":

dfa\_state = stateA(str[i+1])

elif dfa\_state == "b":

dfa\_state = stateB(str[i+1])

elif dfa\_state == "c":

dfa\_state = stateC(str[i+1])

else:

return 0

if dfa\_state == "a" or dfa\_state == "b" or dfa\_state == "c":

return dfa\_state

else:

return 0

str = ["a", "b"]

if NFA(str) == NFA(str [::-1]):

print("Accepted")

else:

print("Rejected")

Text

Description automatically generated Text

Description automatically generated

Reversed string

## Test cases

#1 Let w = ((axb)xc)xb = (axc)xb = bxb = c

wR = ((bxc)xb)xa = (axb)xa = axa = c output: Accepted

Text

Description automatically generated



#2 Let w = ((bxa)xc)xa = (bxc)xb = axa = c

wR = ((axc)xa)xb = (bxa)xb = cxa = c output: AcceptedText

Description automatically generated



#3 Let w = ((cxb)xa)xb = (bxa)xb = bxb = c

wR = ((bxa)xb)xc = (bxb)xc = cxc = c output:Accepted

Text

Description automatically generated



#4 Let w = axc = b

wR = cxa = c output: Rejected

Text

Description automatically generated

