Karachi Institute of Economics and Technology College

Of Computing & Information Sciences

Project Report

Project Name: Mealy Machine

Group Members

Name	SID
Muhammad Saad	10155
Muhammad Ahris	10099
Areeb Ali	10112

Course Name: Automata 108984 –Friday (spring) 2022

Faculty: Sir Aziz Mahmood Farooqi

CONTENTS

ABSTRACT:

In the theory of computation, a Mealy machine is a finite-state machine whose output values are determined both by its current state and the current inputs. This is in contrast to a Moore machine, whose (Moore) output values are determined solely by its current state. A Mealy machine is a deterministic finite-state transducer: for each state and input, at most one transition is possible.

Project Description:

A Mealy machine is a 6-tuple consisting of the following:

- a finite set of states
- a start state (also called initial state) which is an element of
- a finite set called the input alphabet
- a finite set called the output alphabet

- a transition function mapping pairs of a state and an input symbol to the corresponding next state.
- an output function mapping pairs of a state and an input symbol to the corresponding output symbol.

In some formulations, the transition and output functions are coalesced into a single function

- Mealy machines tend to have fewer states:

  Different outputs on arcs (n^2) rather than states (n). .

- In this project first we make a text file on notepad which contains Country code of different cities.

- Then we take city codes as User input and compare the codes from our text file if code is in text file, then print present in file otherwise not present in file.

- Then finally we made mealy machine according to the give codes by user.

3rd Party Libraries and Purpose which used in your project:

1.**Pandas**:

We use pandas library to analyze data which we created from our text file

2.**Automata:**

Automata is a Python 3 library which implements the structures and algorithms for finite automata, pushdown automata, and Turing machines.

3. **IPython:**

IPython provides a rich toolkit to help you make the most out of using Python interactively. Its main components are:

A powerful interactive Python shell

A Jupyter kernel to work with Python code in Jupyter notebooks and other interactive frontends.

Project meeting with your teacher (goal + achievement):

- In first meeting we search about mealy machine
- Then we have to make text file in notepad which contains city code of different countries and search different libraries used in our project.
- Then we have to implement five tuples of mealy machine.
- Next, we have to search implement example of mealy machine.
- Then, we have to hardcode mealy machine examples according to our data.
- Next take input from user and compare if input is match with text file code than print code present in file otherwise not present in file.
- Last implement mealy machine in form of diagram.

Stack info = (Hardware & software requirement for your project):

- System should have python install and can be run on Jupyter notebook , PyCharm or any other ide for python.
- Modern operating system:

  X86 64-bit CPU(Intel/AMD architecture) 8GB RAM.

Methodology (Explanation of complex code logic or complex variable structure):

- Libraries in python
- Automata library is used to make state diagrams in Python
- Used indexing for list to split a num into different variables
- Open and closing are validated by data structure stack (push and pop)
- 5 tuples of mealy machine studied in class
- Used Lucid chart website to make diagrams for mealy machine

Project Code: (For perfect output run on Jupyter notebook)

```
import IPython.display as display
from PIL import Image
import pandas as pd
df = pd.read_table("D:\\Small Projects\\Automata-Project\\Automata-
Project.txt")
print(df)
```

```
class Mealy(object):

    def __init__(self, states, input_alphabet, output_alphabet, transitions,
initial_state):

        self.states = states
        self.input_alphabet = input_alphabet
        self.output_alphabet = output_alphabet
        self.transitions = transitions
        self.initial_state = initial_state
```

```
    def get_output_from_string(self, string):

        temp_list = list(string)
        current_state = self.initial_state
        output = ''
        for x in temp_list:
            output += self.transitions[current_state][x][1]
            current_state = self.transitions[current_state][x][0]

        return output

    def convert_to_moore(self):
        moore_transitions = {}
        temp_list = []
        moore_output_table = {}
        moore_initial_state = self.initial_state
        for x in self.transitions.keys():
            for a in self.input_alphabet:
                temp_list.append(self.transitions[x][a])

        temp_list_2 = []
        for x in temp_list:
            for y in temp_list:
                if x[0] == y[0] and x[1] != y[1]:
                    if x not in temp_list_2 and y not in temp_list_2:
                        temp_list_2.append(x)
                        temp_list_2.append(y)

        temp_list_3 = []
        for x in temp_list_2:
            if x[0] not in temp_list_3:
                temp_list_3.append(x[0])

        if self.initial_state in temp_list_3:
            moore_initial_state = self.initial_state +
self.output_alphabet[0]

        for x in temp_list_2:
            for a in self.input_alphabet:
                if self.transitions[x[0]][a][0] in temp_list_3:
                    next_state = self.transitions[x[0]][a][0]
                    output = self.transitions[x[0]][a][1]

                    next_state = next_state + output
                    try:
                        moore_transitions[x[0] + x[1]][a] = next_state
                    except KeyError as e:
                        moore_transitions[x[0] + x[1]] = {}
                        moore_transitions[x[0] + x[1]][a] = next_state

                    if next_state not in moore_output_table.keys():
                        moore_output_table[next_state] = output
```

```
                else:
                    try:
                        moore_transitions[x[0] + x[1]][a] =
self.transitions[x[0]][a][0]
                    except KeyError as e:
                        moore_transitions[x[0] + x[1]] = {}
                        moore_transitions[x[0] + x[1]][a] =
self.transitions[x[0]][a][0]

                    if moore_transitions[x[0] + x[1]][a] not in
moore_output_table.keys():
                        moore_output_table[moore_transitions[x[0] + x[1]][a]]
= self.transitions[x[0]][a][1]

        for x in self.transitions.keys():
            if x not in moore_transitions.keys() and x not in temp_list_3:
                for a in self.input_alphabet:
                    if self.transitions[x][a][0] in temp_list_3:
                        next_state = self.transitions[x][a][0]
                        output = self.transitions[x][a][1]

                        next_state = next_state + output
                        try:
                            moore_transitions[x][a] = next_state
                        except KeyError as e:
                            moore_transitions[x] = {}
                            moore_transitions[x][a] = next_state

                        if next_state not in moore_output_table.keys():
                            moore_output_table[next_state] = output

                    else:
                        try:
                            moore_transitions[x][a] =
self.transitions[x][a][0]
                        except KeyError as e:
                            moore_transitions[x] = {}
                            moore_transitions[x][a] =
self.transitions[x][a][0]

                        if self.transitions[x][a][0] not in
moore_output_table.keys():
                            moore_output_table[self.transitions[x][a][0]] =
self.transitions[x][a][1]

        moore_states = []
        for s in moore_transitions.keys():
            if s not in moore_states:
                moore_states.append(s)

        from automata.fa.Moore import Moore

        moore_from_mealy = Moore(
```

```
            moore_states,
            self.input_alphabet,
            self.output_alphabet,
            moore_transitions,
            moore_output_table,
            moore_initial_state
        )

        print(moore_from_mealy)

    def __str__(self):
        output = "\nMealy Machine" + \
                 "\nStates " + str(self.states) + \
                 "\nTransitions " + str(self.transitions) + \
                 "\nInital State " + str(self.initial_state) + \
                 "\nInital Alphabet " + str(self.input_alphabet) + \
                 "\nOutput Alphabet" + str(self.output_alphabet)

        return output


mealy = Mealy(
    ['a', 'b', 'c', 'd'],
    ['0', '1'],
    ['0', '1'],
    {
        'a': {
            '0': ('d', '1'),
            '1': ('b', '0')
        },
        'b': {
            '0': ('a', '1'),
            '1': ('d', '1')
        },
        'c': {
            '0': ('c', '0'),
            '1': ('c', '0')
        },
        'd': {
            '0': ('b', '0'),
            '1': ('a', '1')
        }
    },
    'a'
)
mealy_2 = Mealy(['q0'],
                ['0', '1'],
                ['0', '1'],
                {
                    'q0': {
                        '1': ('q0', '0'),
                        '0': ('q0', '1')
                    }
```

```
                },
                'q0'
                )
print("============================ Mealy Machine 1
==============================================")
print(mealy_2)
print()

display.display(Image.open('1.png'))



print("============================ Mealy Machine 2
==============================================")
print(mealy)
print()

display.display(Image.open('2.png'))



mealy_m3 = Mealy(['PK-Khi-00000001', 'PK-Isl-00000001', 'PK-lhr-00000001',
'PK-mul-00000001','US-New-00000001','PK-Psh-00001111','invalid code'],
                ['0', '1'],
                ['correct code', 'incorrect code'],
                {
                    'PK-Khi-00000001' : {
                        '0' : ('PK-Khi-00000001', 'correct code'),
                        '1' : ('PK-Isl-00000001', 'incorrect code')
                    },
                    'PK-Isl-00000001': {
                        '0': ('PK-Isl-00000001', 'correct code'),
                        '1': ('PK-lhr-00000001', 'incorrect code')
                    },
                    'PK-lhr-00000001': {
                        '0': ('PK-lhr-00000001', 'correct code'),
                        '1': ('PK-mul-00000001', 'incorrect code')
                    },
                    'PK-mul-00000001': {
                        '0': ('PK-mul-00000001', 'correct code'),
                        '1': ('US-New-00000001', 'incorrect code')
                    },
                    'US-New-00000001': {
                        '0': ('US-New-00000001', 'correct code'),
                        '1': ('PK-Psh-00001111', 'incorrect code')
                    },
                    'US-New-00000001': {
                        '0': ('PK-Psh-00001111', 'correct code'),
                        '1': ('invalid Code', 'incorrect code')
                    },
                    'invalid code': {
                        '0': ('invalid code', 'incorrect code'),
                        '1': ('Invalid Code', 'incorrect code')
```

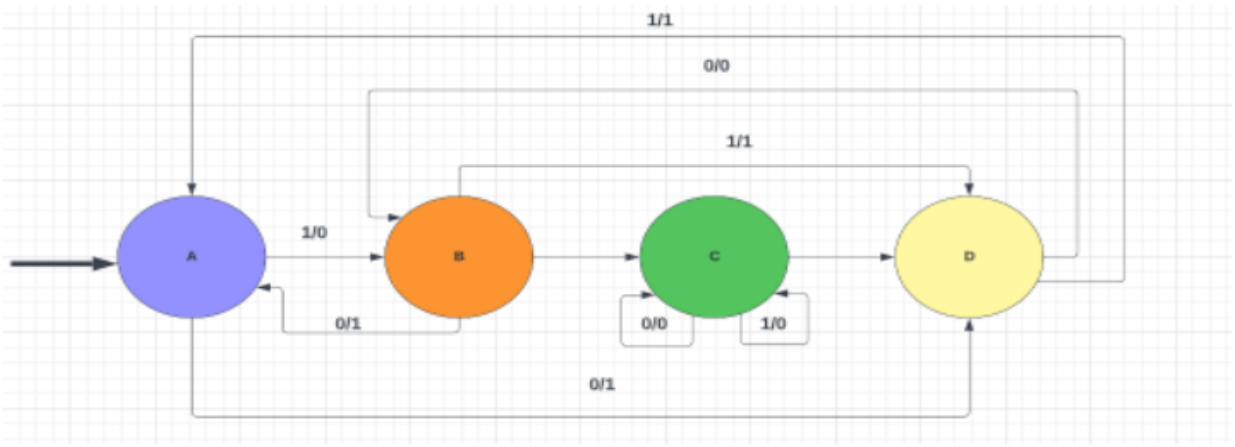
```
                    }
                },
                'PK-Khi-00000001'
                )


print("================================================== Mealy Machine
3 ==================================================")
print(mealy_m3)
print()
display.display(Image.open('3.png'))



C1 = input("Enter the city code 1: ")
C2 = input("Enter the city code 2: ")
C3 = input("Enter the city code 3: ")
C4 = input("Enter the city code 4: ")
C5 = input("Enter the city code 5: ")
C6 = input("Enter the city code 6: ")
print("")
print("City Codes:")
print("Code 1: "+C1)
print("Code 1: "+C2)
print("Code 1: "+C3)
print("Code 1: "+C4)
print("Code 1: "+C5)
print("Code 1: "+C6)
print("")
with open('auto.txt', 'r') as file:
    data = file.read().rstrip()

if C1 in data:
    print(C1 + "  is present in text file")
else:
    print(C1 + " is not present in text file")

if C2 in data:
    print(C2 + "  is present in text file")
else:
    print(C2 + " is not present in text file")

if C3 in data:
    print(C3 + "  is present in text file")
else:
    print(C3 + " is not present in text file")

if C4 in data:
    print(C4 + "  is present in text file")
else:
    print(C4 + " is not present in text file")
```

```
if C5 in data:
    print(C5 + "  is present in text file")
else:
    print(C5 + " is not present in text file")

if C6 in data:
    print(C6 + "  is present in text file")
else:
    print(C6 + " is not present in text file")

print()
mealy_m4 = Mealy([C1, C2, C3, C4,C5,C6,'invalid code'],
                 ['0', '1'],
                 ['correct code', 'incorrect code'],
                 {
                     C1 : {
                         '0' : (C1, 'correct code'),
                         '1' : (C2, 'incorrect code')
                     },
                     C2: {
                         '0': (C2, 'correct code'),
                         '1': (C3, 'incorrect code')
                     },
                     C3: {
                         '0': (C3, 'correct code'),
                         '1': (C4, 'incorrect code')
                     },
                     C4: {
                         '0': (C4, 'correct code'),
                         '1': (C5, 'incorrect code')
                     },
                     C5: {
                         '0': (C5, 'correct code'),
                         '1': (C6, 'incorrect code')
                     },
                     C6: {
                         '0': (C6, 'correct code'),
                         '1': ('invalid Code', 'incorrect code')
                     },
                     'invalid code': {
                         '0': ('invalid code', 'incorrect code'),
                         '1': ('Invalid Code', 'incorrect code')
                     }
                 },
                 'PK-Khi-00000001'
                 )
print("================================================== Mealy Machine
4 ==================================================")
print()
print(mealy_m4)
print()

display.display(Image.open('4.png'))
```

Three test case (input+output):

1)

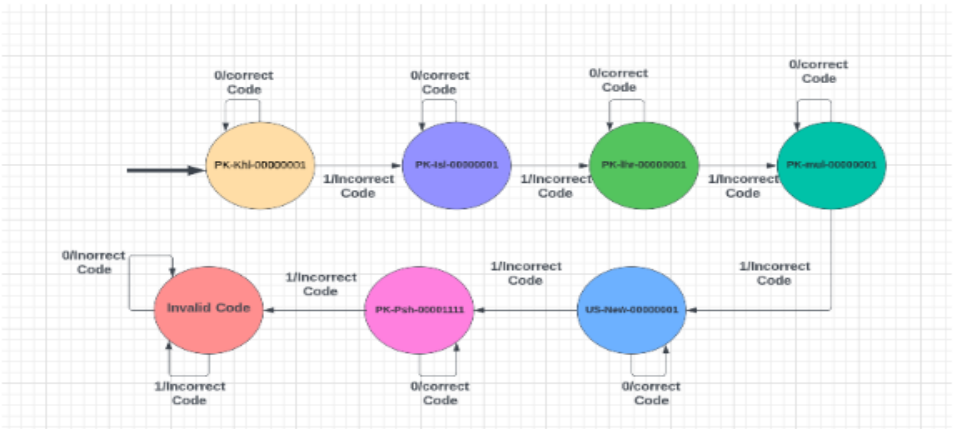
```
============================ Mealy Machine 2 ============================================

Mealy Machine
States ['a', 'b', 'c', 'd']
Transitions {'a': {'0': ('d', '1'), '1': ('b', '0')}, 'b': {'0': ('a', '1'), '1': ('d', '1')}, 'c': {'0': ('c', '0'), '1':
('c', '0')}, 'd': {'0': ('b', '0'), '1': ('a', '1')}}
Inital State a
Inital Alphabet ['0', '1']
Output Alphabet['0', '1']
```



---

2)

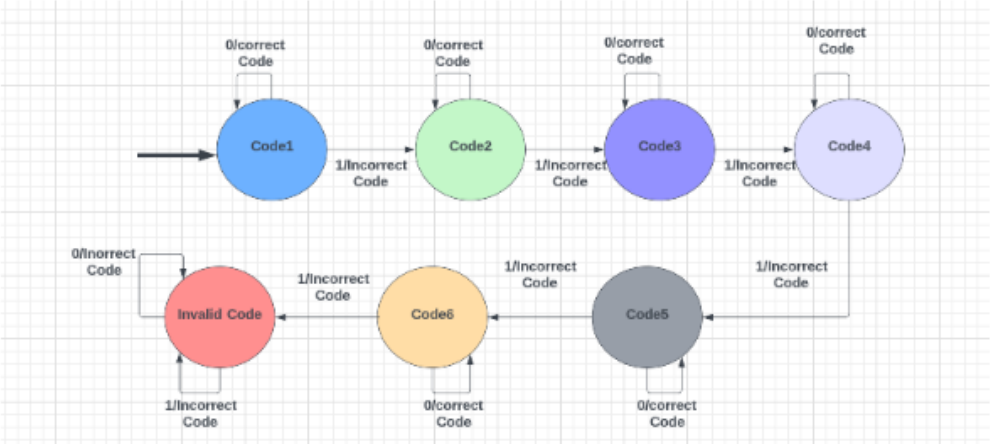
```
========================================================= Mealy Machine 3 ==================================================

Mealy Machine
States ['PK-Khi-00000001', 'PK-Isl-00000001', 'PK-lhr-00000001', 'PK-mul-00000001', 'US-New-00000001', 'PK-Psh-00001111',
'invalid code']
Transitions {'PK-Khi-00000001': {'0': ('PK-Khi-00000001', 'correct code'), '1': ('PK-Isl-00000001', 'incorrect code')}, 'PK
-Isl-00000001': {'0': ('PK-Isl-00000001', 'correct code'), '1': ('PK-lhr-00000001', 'incorrect code')}, 'PK-lhr-00000001':
{'0': ('PK-lhr-00000001', 'correct code'), '1': ('PK-mul-00000001', 'incorrect code')}, 'PK-mul-00000001': {'0': ('PK-mul-0
0000001', 'correct code'), '1': ('US-New-00000001', 'incorrect code')}, 'US-New-00000001': {'0': ('PK-Psh-00001111', 'corre
ct code'), '1': ('invalid Code', 'incorrect code')}, 'invalid code': {'0': ('invalid code', 'incorrect code'), '1': ('Inval
id Code', 'incorrect code')}}
Inital State PK-Khi-00000001
Inital Alphabet ['0', '1']
Output Alphabet['correct code', 'incorrect code']
```



3)

```
Enter the city code 1: PK-Khi-00000001
Enter the city code 2: PK-Khi-00000011
Enter the city code 3: PK-Khi-00000111
Enter the city code 4: PK-Khi-00001111
Enter the city code 5: PK-Khi-00011111
Enter the city code 6: PK-Khi-00111111

City Codes:
Code 1: PK-Khi-00000001
Code 1: PK-Khi-00000011
Code 1: PK-Khi-00000111
Code 1: PK-Khi-00001111
Code 1: PK-Khi-00011111
Code 1: PK-Khi-00111111

PK-Khi-00000001  is present in text file
PK-Khi-00000011  is present in text file
PK-Khi-00000111  is present in text file
PK-Khi-00001111  is present in text file
PK-Khi-00011111  is present in text file
PK-Khi-00111111  is present in text file
```

```
Mealy Machine
States ['PK-Khi-00000001', 'PK-Khi-00000011', 'PK-Khi-00000111', 'PK-Khi-00001111', 'PK-Khi-00011111', 'PK-Khi-00111111',
'invalid code']
Transitions {'PK-Khi-00000001': {'0': ('PK-Khi-00000001', 'correct code'), '1': ('PK-Khi-00000011', 'incorrect code')}, 'PK
-Khi-00000011': {'0': ('PK-Khi-00000011', 'correct code'), '1': ('PK-Khi-00000111', 'incorrect code')}, 'PK-Khi-00000111':
{'0': ('PK-Khi-00000111', 'correct code'), '1': ('PK-Khi-00001111', 'incorrect code')}, 'PK-Khi-00001111': {'0': ('PK-Khi-0
0001111', 'correct code'), '1': ('PK-Khi-00011111', 'incorrect code')}, 'PK-Khi-00011111': {'0': ('PK-Khi-00011111', 'corre
ct code'), '1': ('PK-Khi-00111111', 'incorrect code')}, 'PK-Khi-00111111': {'0': ('PK-Khi-00111111', 'correct code'), '1':
('invalid Code', 'incorrect code')}, 'invalid code': {'0': ('invalid code', 'incorrect code'), '1': ('Invalid Code', 'incor
rect code')}}
Inital State PK-Khi-00000001
Inital Alphabet ['0', '1']
Output Alphabet['correct code', 'incorrect code']
```



Conclusion

However, although a Mealy model could be used to describe the Enigma, the state diagram would be too complex to provide feasible means of designing complex ciphering machines.

Moore/Mealy machines are DFAs that have also output at any tick of the clock. Modern CPUs, computers, cell phones, digital clocks and basic electronic devices/machines have some kind of finite state machine to control it.

Simple software systems, particularly ones that can be represented using regular expressions, can be modeled as Finite State Machines. There are many such simple systems, such as vending machines or basic electronics.

By finding the intersection of two Finite state machines, one can design in a very simple manner concurrent systems that exchange messages for instance. For example, a traffic light is a system that consists of multiple subsystems, such as the different traffic lights, that work concurrently.

Some examples of applications:

- number classification
- watch with timer
- vending machine
- traffic light
- barcode scanner
- gas pumps

We are really thankful to our respected teacher **Sir Aziz Mehmood Farooqi** for providing this opportunity to work on this project.