Computational Maths Project

Deterministic Finite Automaton (AFD)

October 9th, 2017

Enrique Lira Martínez A01023351

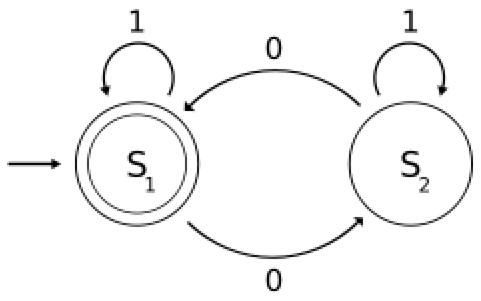
Roberto Alejandro Gutiérrez Guillén A01019608

Campus Santa Fe

What is a Deterministic Finite Automaton?

In the theory of computation, a branch of theoretical computer science, a deterministic finite automaton (DFA)—also known as a deterministic finite acceptor (DFA) and a deterministic finite state machine (DFSM)—is a finite-state machine that accepts and rejects strings of symbols and only produces a unique computation (or run) of the automaton for each input string.

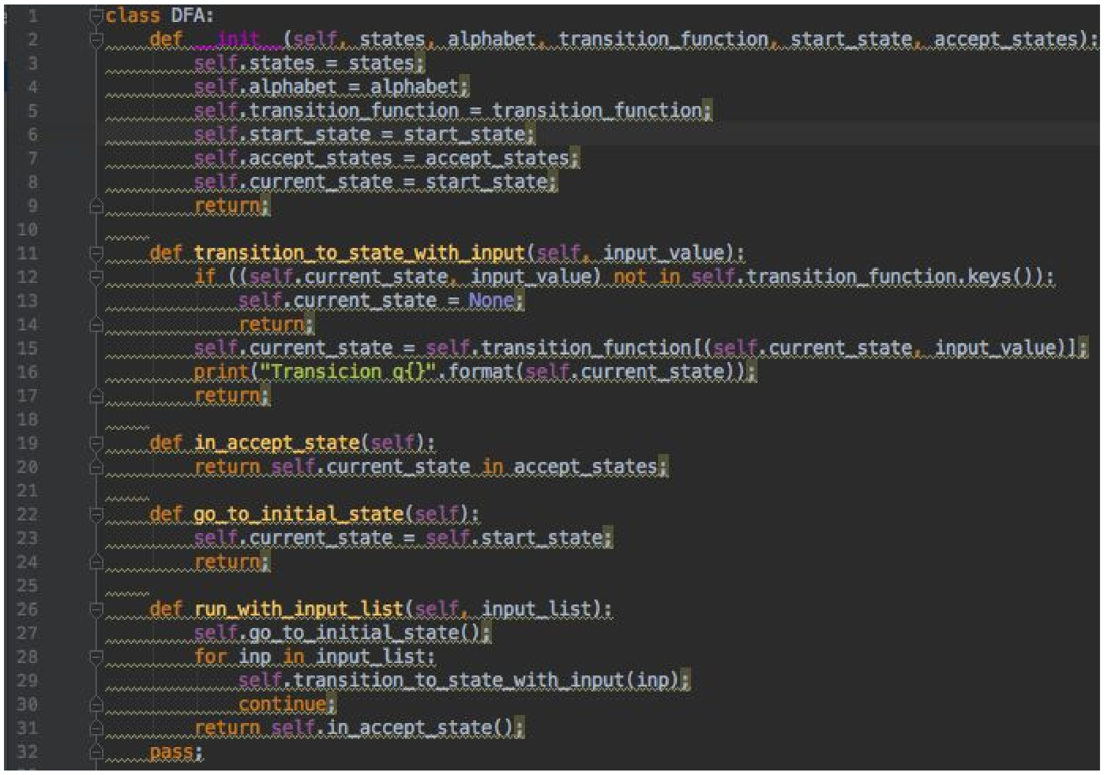
Example:



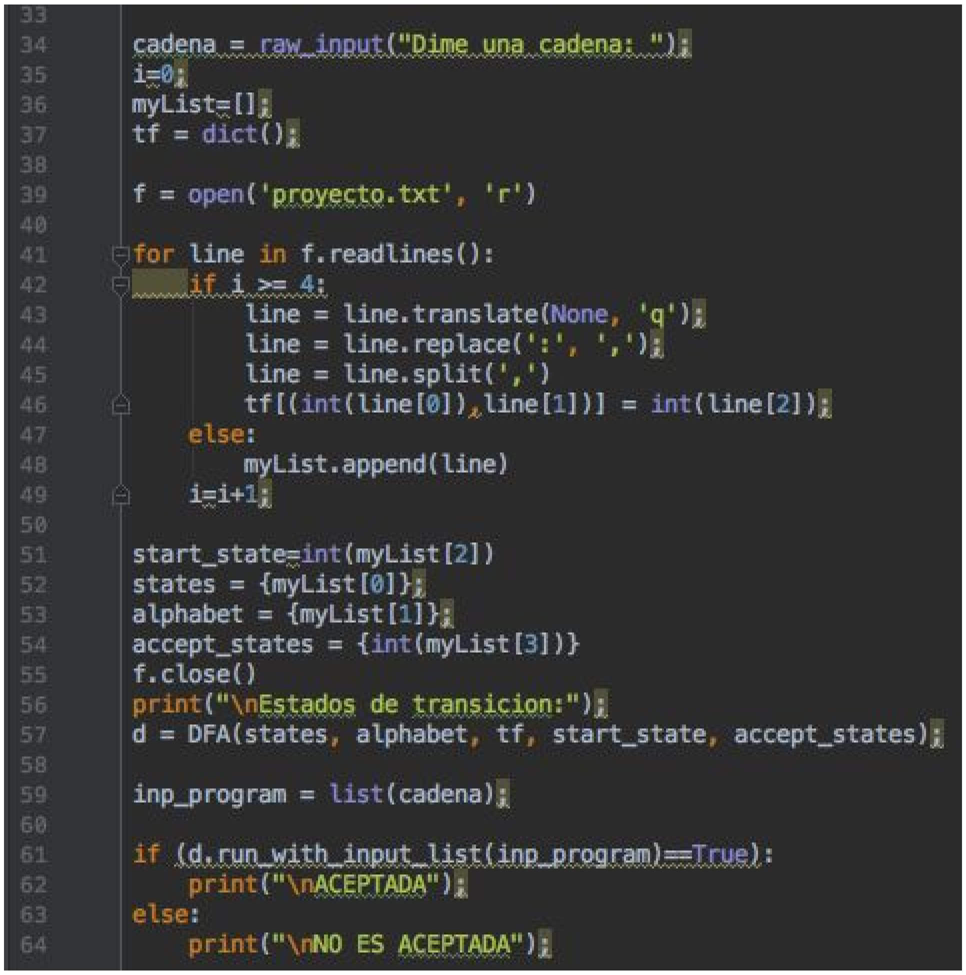
This diagram shows an example of a deterministic finite automaton that accepts only binary numbers. In this example only strings with even ceros are accepted. In this case S1 is the initial and final state.

Explanation of the program

General Structure



```python
class DFA:
    def __init__(self, states, alphabet, transition_function, start_state, accept_states):
        self.states = states;
        self.alphabet = alphabet;
        self.transition_function = transition_function;
        self.start_state = start_state;
        self.accept_states = accept_states;
        self.current_state = start_state;
        return;

    def transition_to_state_with_input(self, input_value):
        if ((self.current_state, input_value) not in self.transition_function.keys()):
            self.current_state = None;
            return;
        self.current_state = self.transition_function[(self.current_state, input_value)];
        print("Transicion q{}".format(self.current_state));
        return;

    def in_accept_state(self):
        return self.current_state in accept_states;

    def go_to_initial_state(self):
        self.current_state = self.start_state;
        return;

    def run_with_input_list(self, input_list):
        self.go_to_initial_state();
        for inp in input_list:
            self.transition_to_state_with_input(inp);
            continue;
        return self.in_accept_state();
    pass;
```



```python

cadena = raw_input("Dime una cadena: ");
i=0;
myList=[];
tf = dict();

f = open('proyecto.txt', 'r')

for line in f.readlines():
    if i >= 4:
        line = line.translate(None, 'q');
        line = line.replace(':', ',');
        line = line.split(',')
        tf[(int(line[0]),line[1])] = int(line[2]);
    else:
        myList.append(line)
    i=i+1;

start_state=int(myList[2])
states = {myList[0]};
alphabet = {myList[1]};
accept_states = {int(myList[3])}
f.close()
print("\nEstados de transicion:");
d = DFA(states, alphabet, tf, start_state, accept_states);

inp_program = list(cadena);

if (d.run_with_input_list(inp_program)==True):
    print("\nACEPTADA");
else:
    print("\nNO ES ACEPTADA");
```

Structure

- **Class DFA**
    - **Constructor DFA**
        - Receives as parameters the following data and loads them as attributes
            - States
            - Alphabet
            - Transition function
            - Start state
            - Accept states
    - **Transition to state with input**
        - Does the transitions between states

        - Prints the transitions between states
    - **In accept state**
        - Validates if the current state is within the accepted states
    - **Go to Initial State**
        - Changes the current state to the start state
    - **Run Input List**
        - Runs the transitions to check if the introduced string is valid or not
        - Returns true if the string is accepted and false if it's not accepted
- **String reader**
    - It reads the input from the user and loads the string to the “cadena” attribute
- **Text file reader**
    - Using open, the text file called "protecto.txt" is loaded onto the variable 'f' using the parameter 'r' for reading only mode.
    - It reads the state's, star state, final state, alphabet and transitions to variables, so they can be sent to the DFA constructor as parameters
    - At the end f.close() closes the file
- **Print Transition States**
    - This section first prints the title of “Transition States”. Afterwards it loads the DFA class and sends to the DFA constructor as parameters the information that was parsed from the text file in the previous section.
- **Print if string is accepted or not Accepted**
    - Inside the if statement the DFA class is executed with the information that was loaded previously from the text file. The function returns a true if the string was accepted or a false if it wasn’t accepted.

Usage Instructions

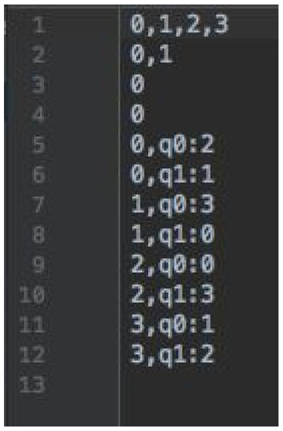
Text file creation:

The text file used as an input to create the automaton needs to follow the next structure exactly as it is stated:

1. In the first line, the set of states needs to be indicated separated by commas
2. The second row indicates the alphabet symbols separated by commas.
3. The third row indicates the initial state

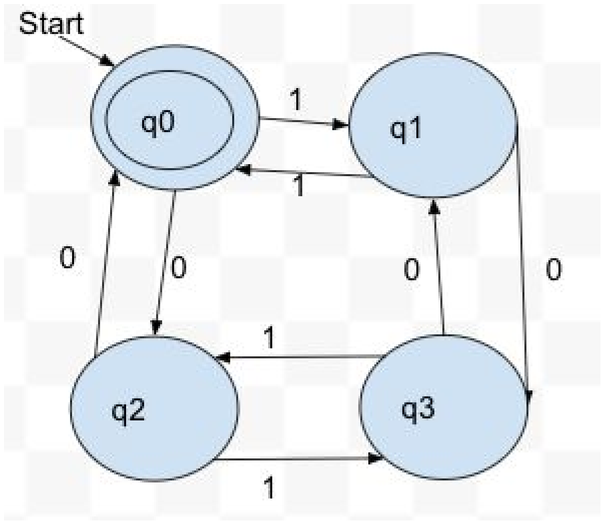
4. The fourth row indicates the final state
5. The next rows after the fourth one indicate the evaluation of the transition function in the following format: initial state, symbol: final state
    a. Each transition needs to be specified in separate lines

*The states need to be referenced with only numbers. An example of a text file that is accepted by the program can be found below.



```
0,1,2,3
0,1
0
0
0,q0:2
0,q1:1
1,q0:3
1,q1:0
2,q0:0
2,q1:3
3,q0:1
3,q1:2
```

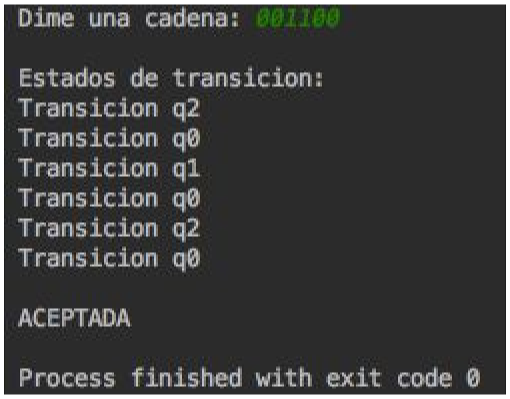
Example:



The automaton above, is the one used in the program. This automaton is programmed in the text file, so when the program reads the text file de DFA is created in the class and the strings that are entered by the user can be tested whether they are valid or not in this DFA.

Accepted string example:

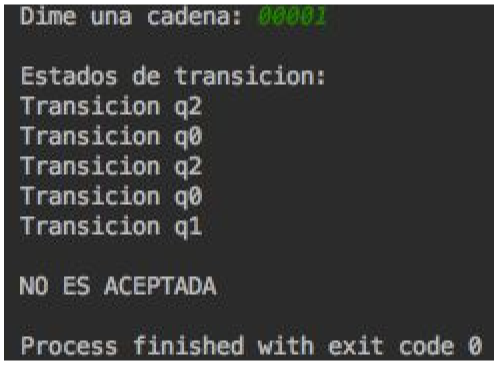
In this example we can appreciate that the program accepted the following string “001100” this is due to the fact that it has an even number of 1’s and 0`s, although there is a different number of 1’s and 0’s both of them are even



```
Dime una cadena: 001100

Estados de transicion:
Transicion q2
Transicion q0
Transicion q1
Transicion q0
Transicion q2
Transicion q0

ACEPTADA

Process finished with exit code 0
```

Denied string example:

In the image below we can appreciate that the program printed the states the string went through, also it prints that it's not accepted, as the number of 1’s is not even.



```
Dime una cadena: 00001

Estados de transicion:
Transicion q2
Transicion q0
Transicion q2
Transicion q0
Transicion q1

NO ES ACEPTADA

Process finished with exit code 0
```

References

- Hopcroft, John E.; Motwani, Rajeev; Ullman, Jeffrey D. (2001). *Introduction to Automata Theory, Languages, and Computation* (2 ed.). Addison Wesley. ISBN 0-201-44124-1. Retrieved 19 November 2012.