A MINI-PROJECT REPORT

ON

‘DFA and its applications’

Submitted to

Dr. Shyamapada Mukherjee

BY:

Ashutosh Varshney (18-1-5-089)

For end-semester exams (4th semester)

Course:CS-204

BATCH OF 2018-2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, SILCHAR(ASSAM)

TABLE OF CONTENT

PROJECT SYNOPSIS

The sole focus of this report was to find out how various DFAs are used in real-world applications. Started with a simple one and jumped to a not-so-sophisticated one, it has been explained by taking few examples. A brief and comprehensible explanation of every example has been given. Basic working of DFA has also been discussed.

- ## Deterministic Finite Automata:

For a given input symbol to which next state it will move to can be determined, hence the name deterministic automata. Number of states being finite gives it the name finite automata and on combining both we get the Deterministic Finite Automata (DFA).

Formal definition:

A **DFA** can be denoted by a 5-tuple (**Q, ∑, δ, q0, F**) where,

1. **Q** is a set of states. The number of states are finite.

2. ∑ is a set of symbols called the alphabet. They are finite in number.

3. **δ** is the transition function. Here it maps **Q** × ∑ → **Q**

4. **q0** is the initial state from where any input is first given **(q0 ∈ Q)**.

5. **F** is a set of final state/states of **Q (F ⊆ Q)**.

State diagram of DFA:

We denote,

- States by circles.
- Transitions by arcs.
- Initial state by incoming arc.
- Final states or accepting states by two circles.

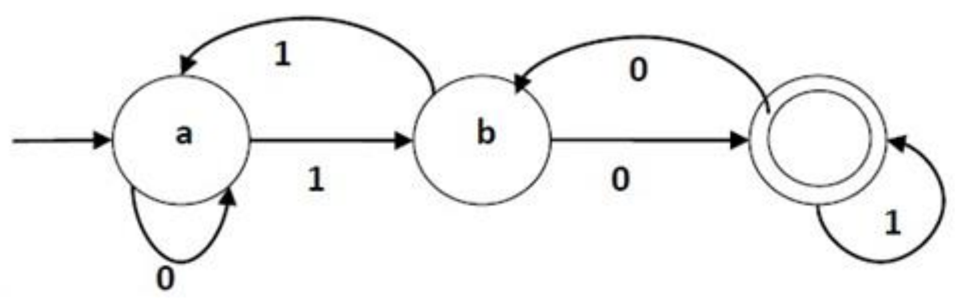
Example:

DFA:

- Q = {a, b, z},
- ∑ = {0, 1},
- q0 = {a},
- F = {z},

Transition function:

Current State	**Input 0**	**Input 1**
a	a	b
b	z	a
z	b	z



State diagram

- **<u>Applications:</u>**

  DFAs have strong mathematical(theory) connection which has framed its development and use over the time. Many simple and sophisticated machines, oftentimes, are designed and implemented using DFAs. Its practical applications include: Almost all compilers, language-processing systems, text processors, etc.

  Here is a short and non-exhaustive list of DFA applications:

- Vending Machines
- Traffic Lights
- Video Games
- Lexer
- Regular Expression Matching
- CPU Controllers
- Protocol Analysis
- Natural Language Processing
- Speech Recognition

NOTE: We are not going to discuss about every application given in the list in detail. We will be discussing only few.

1. Vending Machines

Figure 1 presents the DFA which illustrates the operation of the vending machine which accepts dollars and quarters, and charges $ 1.25 for a soda. If the machine gets at least $ 1.25, which corresponds to the blue areas in the diagram, will allow user to choose soda. Self-loops represent a neglected input: the machine will not dispense a soda up to a minimum of $ 1.25, and it won't input much cash if it already has received greater than or equal to $ 1.25.

Its formal explanation goes like this:

1. Q = {$0.00, $0.25, $0.50, $0.75, $1.00, $1.25, $1.50, $1.75, $2.00}
2. ∑ = {$0.25, $1.00, select}
3. δ, the transition function.
4. q0 = $0.00 is the initial state.
5. F = Φ is the set of final state/states.

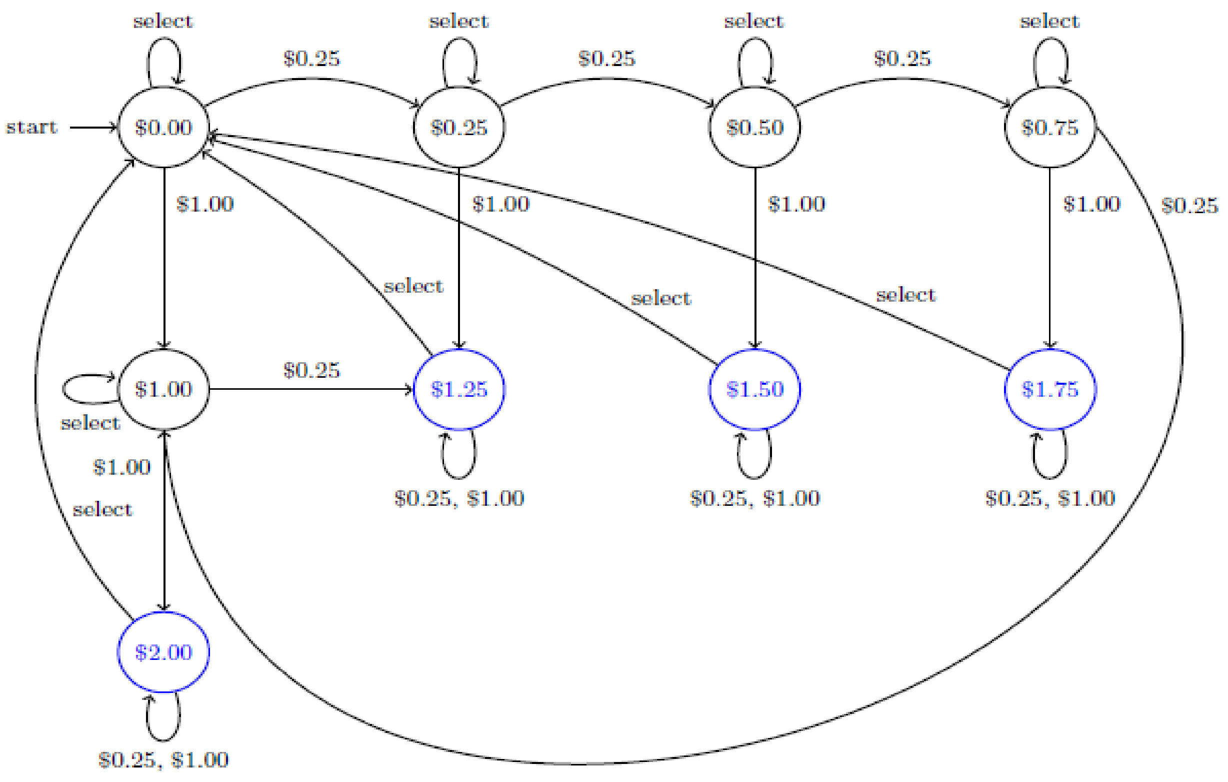


Figure 1: Vending Machine State Diagram

2.Pac-Man’s ghost

Video game characters controlled by the computer can be a good example of the use of finite state machines. Machine controlled characters’ behavior vary according to the various circumstances. Character’s behavior corresponds to the different machine’s states. These changes in states can be observed in the state diagram. Simple game characters which exhibit state-based behavior can be controlled using state machines.

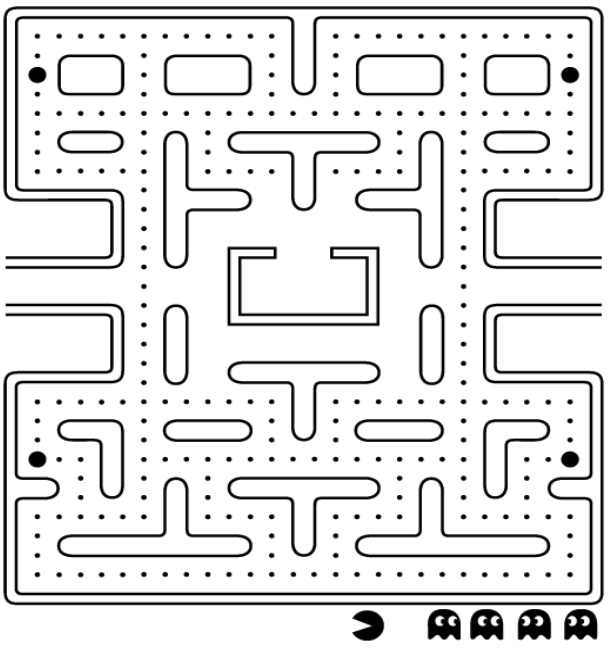


Figure 2: Screenshot of a Pac-Man clone
Image Source: Here

In figure 2, a screenshot is given of a very popular retro game Pac-Man, rules are very simple (for those who are unaware of the game-play). Player needs to move through the maze, eating pellets and avoiding getting caught by the ghosts who chase him throughout the maze. Only temporary special ability the player has is, it can eat the ghosts if it gets to consume the power pellet, consequently ghosts behavior change and opposite to their normal behavior they start avoiding Pac-Man.

The ghosts in Pac-Man exhibit four different behaviors:

1. Randomly roam throughout the maze.
2. Chase Pac-Man when they both meet the eye.
3. Flee Pac-man when it consumes the power pellet.
4. Return to the central base to regenerate.

Thus, these four behaviors become the key to the four-state DFA.

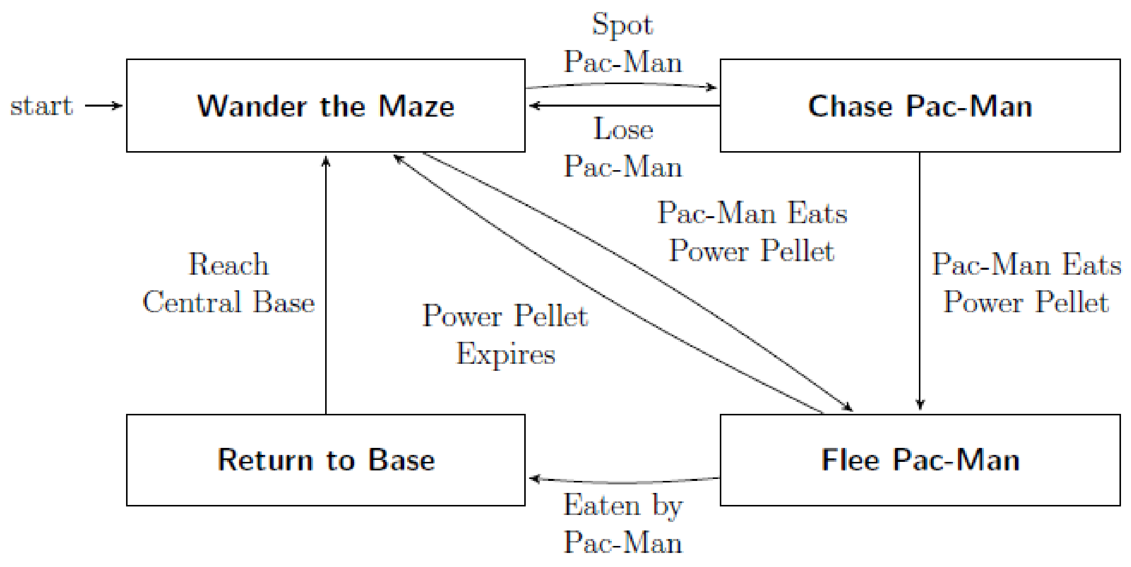


Figure 3: Behavior of a Pac-Man Ghost

3.DFAs in compilers

Compiler working starts as a lexer in every programming language. The lexer checks through the program file and produces a series of tokens.
For example, if c++ code contains the line:

```
1.   cout  <<   "TOC mini-project" <<  endl;
```

then lexer produces something like this:

```
1.   IDENTIFIER     cout
2.   LSHIFT         <<
3.   STRING          "TOC mini-project"
4.   LSHIFT          <<
5.   IDENTIFIER      endl
6.   SEMICOLON     ;
```

DFA is used in a lexer to read through every file character wise resulting into generating tokens.

CONCLUSION

1. DFA has a very broad theoretical aspect.
2. Any system maintaining an internal definition of state can be represented by DFAs.
3. DFA requires a constant amount of memory no matter how lengthy the input is.
4. DFA makes the behavior of the system it represents easy to understand owing to its one input fixed transition and other features.