**CSE208 D2**

**Automata of an elevator**

**Report by**

Team members:

D. SHASHANK - 13BCE1036

K. GOURAV - 13BCE1068

M. V. SHASHANK - 13BCE1081

*A project report submitted to the*

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

*Under the guidance of*

**Prof. B. V. A. N. S. S. Prabhakar Rao**

*In partial fulfillment of the award for degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE ENGINEERING**



**VIT UNIVERSITY, CHENNAI**

**Vandalur – Kelambakkam Road**

**Chennai – 600 127.**

**NOVEMBER 2014**

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof. B. V. A. N. S. S. Prabhakar Rao**r, School of Computer Science and Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. L. Jagannathan,** Dean of the School of Computer Science and Engineering, VIT University, for extending the facilities of the School towards our project and for his unstinting support.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

**Shashank. D Gourav. K Shashank. M. V**

Postulates:

* Elevator is used in different areas and aspects in our day to day life.
* Due to the advancement of technology, it is said by every person in this world today that whenever a more than three storied building is seen, there should be a lift (elevator) in this building.
* But this lift cannot be constructed as easy as said.
* As every electronic machine works on a particular principle, even lift has an interior design and a principle that it works on.
* This principle can be designed with the help of automata or the design of a lift is a derived automata.
* Every lift is not associated with a human brain so that it comes and goes to whichever floor whenever we want but it is associated with an automata which helps the elevator to function in a righteous manner.
* Assuming the initial state of the lift to be present in ground state, starting state is initialized as ground state and the number of final states as no floors.
* There need to be a provision such that on excess load, the automata should remain in the same state until and unless the load is decreased.
* There is no rule that at a time only a single input is obtained and hence the input is taken as a string.
* At each state different input strings are received and the strings are to be sorted in order that the lift functions properly.
* If the input has repeated strings, while the sorting mechanism is in action, the repetitions have to be truncated.
* At each acceptance state, there need to be two states – one for door opening and another for door closing.
* At a particular state if the previous received input contains a string with states greater than that state and in that state, if another input is received with states lesser than the state itself, while sorting, the lift should consider only the states above the state and on exhausting all the states above it, then the below states should be attended in descending order sorting mechanism.
* At a given state, if the string doesn’t contain succeeding states, then it has to go to the preceding states in descending order.
* At a given state, if the current state is also present as input, it has to go to the door opening state.
* After a particular state is attended, the state must be eliminated from the list.

Let's imagine an elevator that serves two floors. Inputs are calls to a floor either from inside the elevator or from the floor itself. This makes three distinct inputs possible, namely:

0 - no calls

1 - call to floor one

2 - call to floor two

The elevator itself can be going up, going down, or halted at a floor. If it is on floor it could be waiting for a call or about to go to the other floor. This provides us with the six states shown in figure 2 along with the state graph for the elevator controller.

W1 Waiting on first floor

U1 About to go up

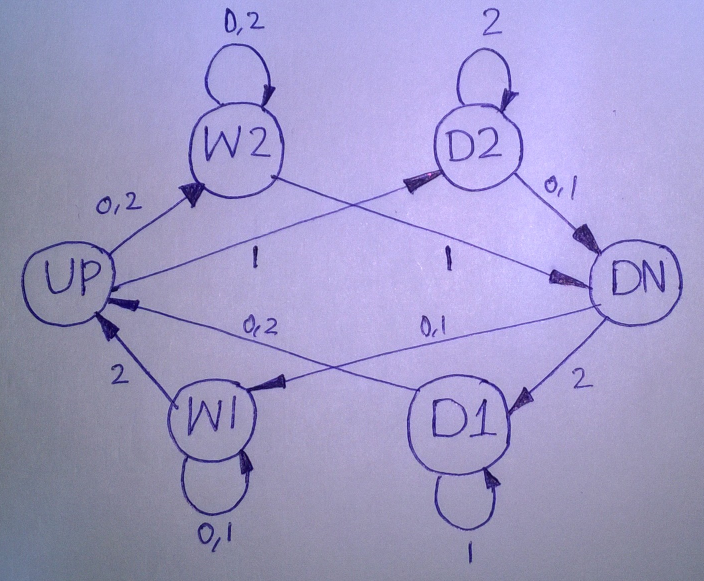
UP Going up

DN Going down

W2 Waiting on second floor

D1 About to go down from first floor

D2 About to go down from second floor



There are no acceptance and rejectance states because it does not makes sense unless and until issues like power failure, overloading etc. are taken into consideration.

Designing an automata of an elevator is simple but the motive of this project is to bring awareness among ourselves about the knowledge of automata to its exposure in day to day life.

This is a normal automata but a peculiar automata should be designed in such a manner that all the above stated rules are satisfied.

Every automata has a spring balance or a weight estimator attached to it which determines the weight of the elevator box. The elevator will be attached to an elongated wire on which the elevator moves up and down. Hence when the weight of the elevator box equals to the force due to the tension inside the string, a signal is sent that the elevator is overloaded and hence a rejectance state is set such that it does not allow the elevator to reach the door closing state unless the load is decreased.

So, to control the elevator we need three interfaces namely elevator control, elevator body and user interface.

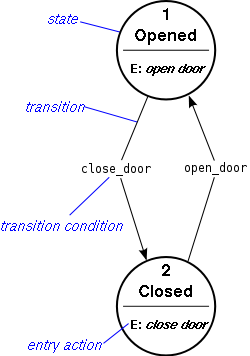
User interface is the one which is the only thing that the user can access and is linked with the elevator control.

User interface contains buttons which give input to the elevator.

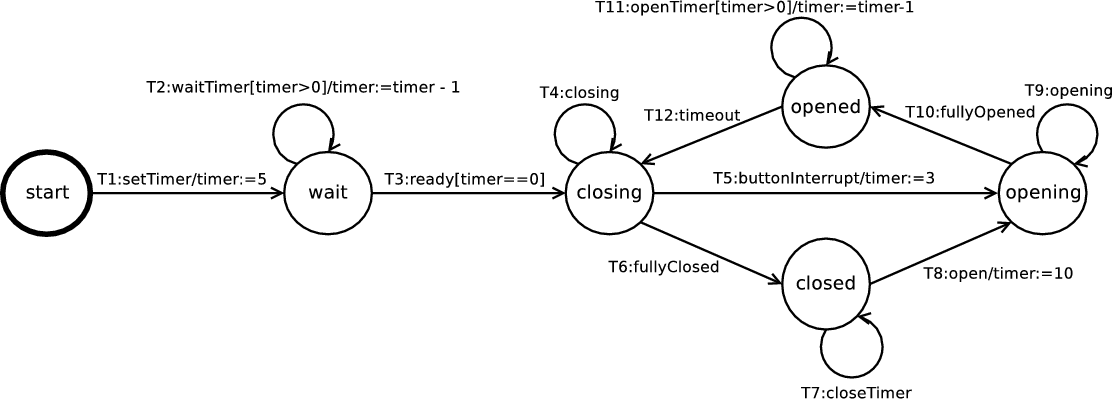
inside the elevator outside the elevator

Inside the elevator there is a board of all the floor buttons each of which indicates a destination folder. Outside the elevator there are two direction buttons on each floor which call for service, one for up and the other for down except on the first and top floors in which only one button either up or down is required. Any button can be pushed at any time. Any floor button will be on from the time it is pushed until the elevator stops at the floor. Any direction button will be on from the time it is pushed until the elevator stops at the floor with the same direction. A more complex elevator will have door open and close buttons, alarm emergency button etc. which for simplicity is not considered. The basic actions of an elevator are move-up or move-down one floor, serve a floor i.e. stop at the floor, open and close the door and stay idle until further input is given. These actions are called basic because they do not mix up within themselves. For example, it is not possible to have an action as “open, move-up, close…”

The door opening and closing states inside the elevator will be as follows:



The elevator door opening and closing states outside the elevator are as follows:



The sorting mechanisms are also implemented using a different finite control which comes under elevator body.

The combination of all the above designs is the overall design of the elevator.

Suppose a person presses a button for the lift but later changes his mind and moves away, in order that the lift should not stop at that floor, along with up and down, let us put another button such that while leaving, if the person presses the button, the lift will ignore the input.

A C program is implemented to design an automata and find whether a string is accepted or not.

Source Code:

#include<stdio.h>

#include<conio.h>

int ninputs;

int check(char,int ); //function declaration

int dfa[10][10];

char c[10], string[10];

int main()

{

int nstates, nfinals;

int f[10];

int i,j,s,final;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nTeam Members:\n1. D. Shashank\n2. K. Gourav\n3. M. V. Shashank\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n\t\tDeterministic Finite Automata\n");

printf("\nEnter the number of states that your dfa consist of: ");

scanf("%d",&nstates);

printf("\nEnter the number of input symbol that dfa have: ");

scanf("%d",&ninputs);

printf("\nEnter the input symbols\t");

for(i=0; i<ninputs; i++)

{

printf("\n\n %d input\t", i+1);

printf("%c",c[i]=getch());

}

printf("\n\nEnter the number of final states\t");

scanf("%d",&nfinals);

for(i=0; i<nfinals; i++)

{

printf("\n\nFinal state %d : q",i+1);

scanf("%d",&f[i]);

}

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n\ndefine transition rule as (initial state, input symbol ) = final state\n");

for(i=0; i<ninputs; i++)

{

for(j=0; j<nstates; j++)

{

printf("\n(q%d , %c ) = q",j,c[i]);

scanf("%d",&dfa[i][j]);

}

}

do

{

i=0, final=0, s=0;

printf("\n\nEnter Input String.. ");

scanf("%s",string);

while(string[i]!='\0')

if((s=check(string[i++],s))<0)

break;

for(i=0 ; i<nfinals ; i++)

if(f[i] ==s )

final=1;

if(final==1)

printf("\n valid string");

else

printf("invalid string");

printf("\nDo you want to end <y/n> ? ");

}

while(getch()=='y');

getch();

}

int check(char b,int d)

{

int j;

for(j=0; j<ninputs; j++)

if(b==c[j]){

return(dfa[j][d]);}

return -1;

}

Output:

