3rd International Conference on Computing: Communication, Networks and Security (IC3NS-2018)
Volume: 4 Issue: 3
ISSN: 2454-4248
35 – 38

A Turing Machine Test Bench

Aastha Gupta, Aashi Jain, Aastha Sharma
College of Engineering and Technology, Mody University of Science and Technology, Lakshmangarh (Rajasthan), 332311, India.
aasthag1811@gmail.com, jainaashi1998.ja@gmail.com, aasthacse@gmail.com
+91 8168377718, +91 9457197007, +91 7597741745

Abstract: A turing machine is a machine which can solve almost any problem. If for a problem there exists an algorithm, then there surely exists a turing machine. A turing machine test bench is a platform where one can implement his turing machine and check it for various inputs. The major problem that occurs during the testing of a turing machine is that the programmer is not able to understand where the problem has occurred and why his machine is not working properly. A common turing machine compiler only lets you know whether the machine is accepting the input or not. To overcome this problem we have introduced the concept of instantaneous description.

We have used java to implement our test bench as java is a platform independent language. We have used Eclipse Mars 2.0 IDE.

*****

I. INTRODUCTION

A turing machine is a hypothetical machine that was thought of by the mathematician Alan Turing in the year 1936. Despite its simplicity, the machine can simulate any computer algorithm, no matter how complicated it is.

TURING MACHINE TEST BENCH

A turing machine test bench is a platform to configure user-defined machines and test them. We have implemented the same along with some in-built machines. We have also shown instantaneous description of each input symbol in input string which helps user to understand how an input string is computed.
Jflap [3] is one such software where one can construct his own turing machine using drag and drop method and can check it by giving some input string. But, the problem with jflap is that it does not tell where the problem has occurred and so we have tried to overcome this shortcoming.

II. METHODS

There are some in-built machines which the user can run and can test with input strings. The mentioned in-built machines are a string followed by same string, 2's complement and addition of two unary numbers. In addition to this, the user can configure his own machine be it acceptor or transducer. The user can configure his machine and then check it using input strings.
The block diagram of the test bench is as follows-

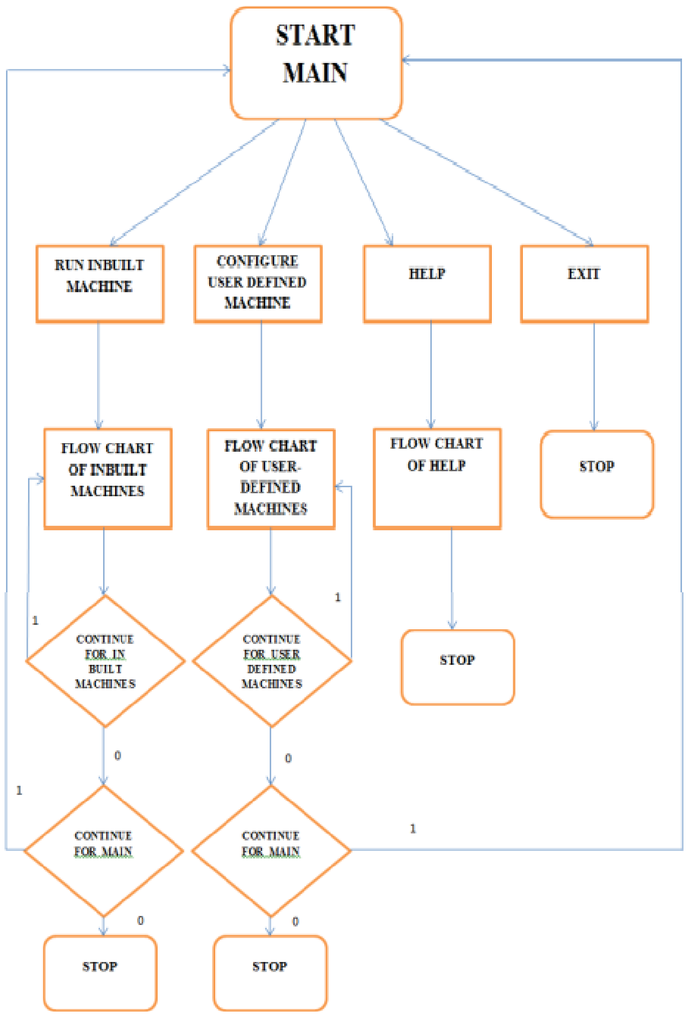
START MAIN
RUN INBUILT MACHINE
CONFIGURE USER DEFINED MACHINE
HELP
EXIT
FLOW CHART OF INBUILT MACHINES
FLOW CHART OF USER-DEFINED MACHINES
FLOW CHART OF HELP
STOP
CONTINUE FOR IN BUILT MACHINES
CONTINUE FOR USER DEFINED MACHINES
STOP
CONTINUE FOR MAIN
CONTINUE FOR MAIN
STOP
STOP
1
1
0
0
1
1
0
0

Fig A: Block Diagram of the Main Menu

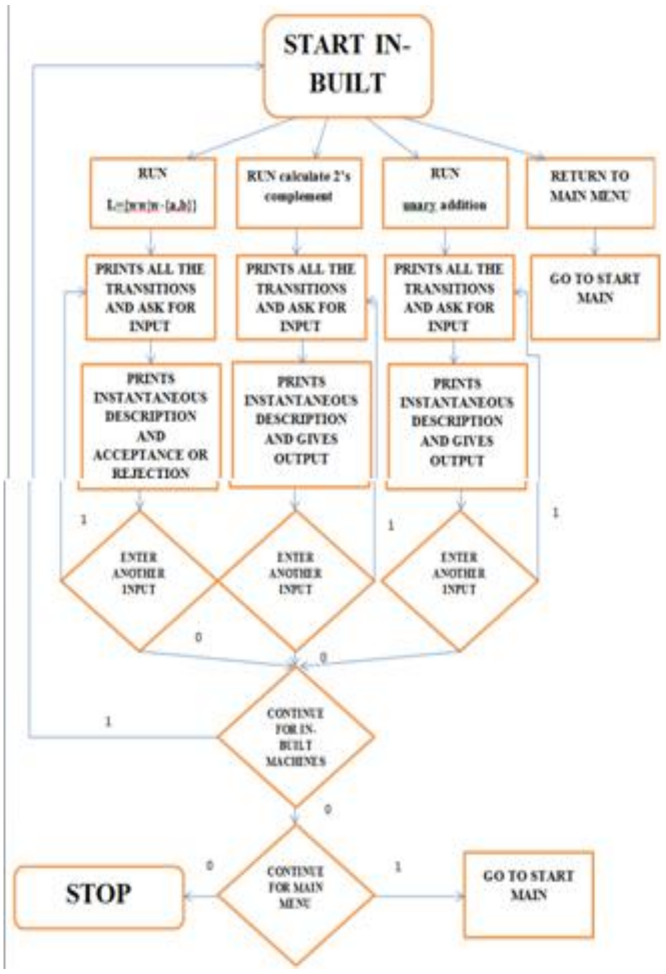
START IN-BUILT
RUN L={w\|w-{a,b}}
RUN calculate 2's complement
RUN unary addition
RETURN TO MAIN MENU
PRINTS ALL THE TRANSITIONS AND ASK FOR INPUT
GO TO START MAIN
PRINTS INSTANTANEOUS DESCRIPTION AND ACCEPTANCE OR REJECTION
PRINTS INSTANTANEOUS DESCRIPTION AND GIVES OUTPUT
ENTER ANOTHER INPUT
1
0
CONTINUE FOR IN-BUILT MACHINES
CONTINUE FOR MAIN MENU
STOP
GO TO START MAIN

Fig B: Block Diagram of the In-built Machines Menu

The above diagram shows the in-built machines option. This diagram shows the complete working of the in-built machines menu. After complete execution of the in-built machine, the user will be asked if he wants to continue for another machine or not. If the user presses 1 then he will be redirected to the beginning of the in-built machine menu, else he will be redirected to the main menu.

The below block diagram shows the working of the user-defined machines option. The diagram completely describes step-by-step flow of the control of the program and also shows the direction of the program according to the inputs of the user.

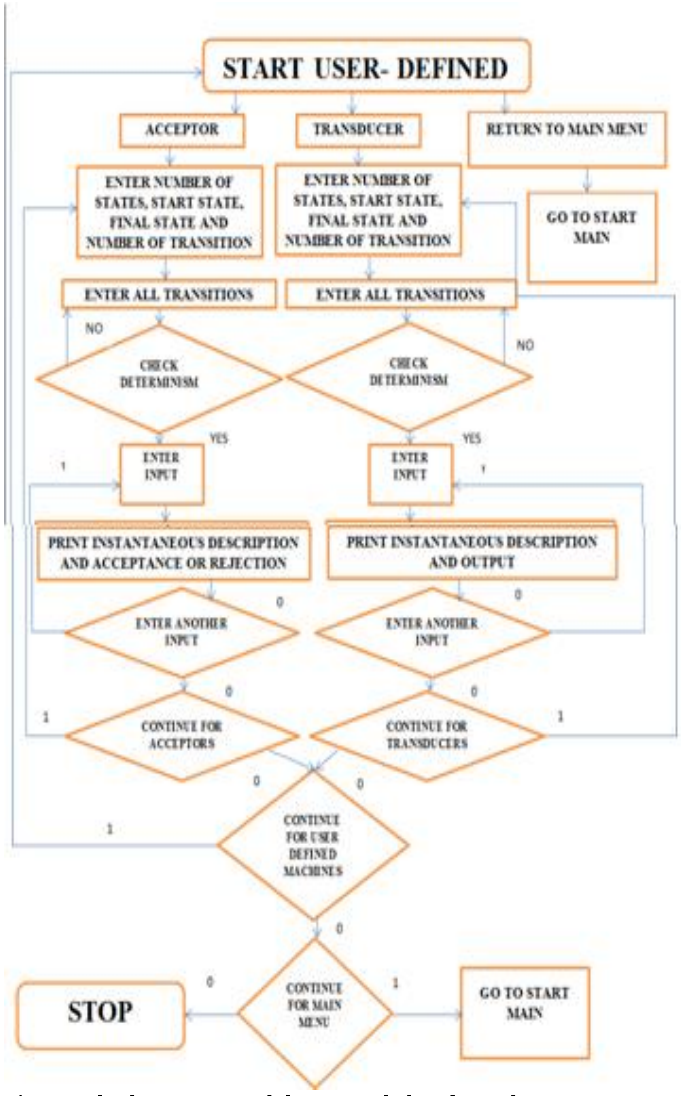

Fig C: Block Diagram of the User-defined Machines Menu

The main problem that users face is they are not able to check where the fault has occurred in the machine. To overcome this problem we have given the instantaneous description for each input symbol so that the user can check where the problem has occurred.

In addition to this we have also added the feature of checking if the user has entered any non-determinism in the machine. As we know that turing machines are deterministic, any non-determinism can result in infinite looping of the machine or giving improper result. Therefore our program checks determinism and asks the user to re-enter the machine if non-determinism is found.

We have also added one more possibility for the read-write head of the input tape. Generally the read-write head moves either left or right, but in addition we have added a stay option, i.e., the read-write head can stay at its position.

Initially the user views the program as a menu-driven program asking to choose an option among running in-built machines, configure any user-defined machine, exit or help. The in-built machines option leads to another multi options

which contain run the ww machine, calculate 2’s complement of a number or to do unary addition of two numbers. The user can choose any option among them and run the corresponding machine by giving an input string. The user will get the instantaneous description of the input and the final result.
If the user chooses to configure his own machine then he has to choose whether to configure an acceptor machine or a transducer machine. Then the user has to enter the start state, final states and all the transitions of the turing machine. The program stores this machine in the form of a 2-D array. Then the user has to enter an input string which results in the instantaneous description along with the result or acception/rejection.

III. EXAMPLE

Let us consider an example to understand the working of this project. Let the user wants to test a machine which calculates the 1’s complement of a number. Initially the user will get the MAIN MENU on the screen. Then he has to select the option “Configure user-defined machines” i.e., option 2.
Then the user will get another menu that is the USER-DEFINED MACHINES menu in which there will be three options of acceptors, transducers, and return to main menu. As the user has to make a transducer machine, he will select the option 2.
After that the user has to enter the machine. First the user will be asked to enter the number of states in the machine. There are four states in this machine so the user will enter 4. Next he has to give the start state i.e. 1 and then the number of final states. As this machine has only one final state so the user will enter 1. Then he will be asked to enter the final states. So the user will enter 4 there. Next we have to give all the transitions. Number of transitions will be asked which is 5 in this case. Now we have to enter all the transitions. It will be done in the following manner:

TRANSITION NUMBER 1
Start state: 1
End state: 2 Transition: 1,0,r/0,1,r

TRANSITION NUMBER 2
Start state: 2
End state: 2 Transition: 0,1,r/1,0,r

TRANSITION NUMBER 3
Start state: 2
End state: 3 Transition:
#,#,l

TRANSITION NUMBER 4
Start state: 3
End state: 3 Transition: 1,1,l/0,0,l

TRANSITION NUMBER 5
Start state: 3
End state: 4 Transition:
#,#,r

In this manner the whole machine has been entered. Now the machine will be displayed in the form of a 2-D array. This machine will be displayed in the following manner:

```
q1            q2       q3       q4
q1      -    1,0,r/0,1,r   -       -

q2      -    0,1,r/1,0,r   #,#,l   -
q3      -       -    1,1,l/0,0,l   #,#,r q4          -
        -       -   -
```

Now the machine will be checked. In order to do this the user will be asked to enter some input. Suppose the user gives the input string as 1011. The user
will get the instantaneous description of the input string for each input symbol. This will appear in the following manner:

```
########## q1
1011####################################
##########0 q2
011####################################
##########01 q2
11####################################
##########010 q2
1####################################
##########0100 q2
####################################
##########010 q3
0####################################
##########01 q3
00####################################
##########0 q3
100####################################
########## q3
0100####################################
######### q3
#0100####################################
```

And the final output will be:

```
##########0100##################################
##
```

In this manner the final output is obtained. After that the user will be asked if he wants to enter any other input for the same machine. If yes then press 1 otherwise press 0. As we do not want to enter any other input for the same machine so we will press 0. Now the user will be asked if he wants to enter any other transducer machine. Press 1 if yes otherwise press 0. So the user will press 0 as we do not want to enter any other machine now. Similarly now the user will be asked if he wants to continue for user-defined machine. If the user presses 0 then he will be asked if he wants to continue for the main menu. If the user presses 0 then the program will stop; otherwise the user can continue with the main menu. In this manner the whole program is executed.

IV. CONCLUSION

The main highlighting features in this project are-

1. Flexibility in usage of symbols, i.e. proficient in allowing any input symbol.
2. Code is not specific to any language. Due to the inclusion of simple for loops and switch cases, it can very easily be translated into any language.
3. Users can configure both transducers and acceptors.
4. Apart from the standard Turing Machine, user can also configure a Turing Machine with stay option.
5. User can follow step by step computation through instantaneous description of each input symbol.
6. Non-determinism can be identified and later rejected on the lines of the fact that a standard Turing Machine is a deterministic.

V. FUTURE SCOPE

The future scope of this project is that one can modify it to have a feature in which user can design his machine by simple point and click design which will make the project more user friendly.

REFERENCES

[1]. Peter Linz (5th ed.). (2011) *An Introduction to Formal Languages and Automata*, Jones and Bartlett Learning

[2]. Hopcroft, Motwani, and Ullman (3rd ed.). (2006) *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing Company

[3]. Building a Nondeterministic Finite Automaton Retrieved February 5, 2018 from http://www.jflap.org/tutorial/fa/createfa/fa.html