2022 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES) | 978-1-6654-8004-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/CISES54857.2022.9844404

Implementation of Tic Tac Toe Game using Multi-Tape Turing Machine

Sai Surya Teja Gontumukkala, Yogeshwara Sai Varun Godavarthi, Bhanu Rama Ravi Teja Gonugunta, Supriya M.*
Department of Computer Science and Engineering
Amrita School of Engineering, Bengaluru
Amrita Vishwa Vidyapeetham, India
saisurya029@gmail.com, gysaivarun31@gmail.com, ravitejagonugunta150@gmail.com, m_supriya@blr.amrita.edu

***Abstract*— Automaton is a model of computation based on a theoretical machine and is composed of one or more states. The application of automata concepts to the game design has proved to be an efficient way of solving the problem. This paper proposes to use Turing machine in designing the tic-tac-toe game. Tic Tac Toe game is the most popular two-player board game of all time. Nowadays Tic Tac Toe game is available in the form of many mobile and pc applications. This game is being implemented by using Multi-tape Turing machine. A Multi-tape Turing machine is a variation of the Turing machine which has multiple tapes and results are experimented with using JFLAP.**

Keywords—Tic Tac Toe, Multi-Tape Turing Machine, JFLAP

I. INTRODUCTION

The study of abstract computing devices, also known as abstract machines, is called automata theory. An automaton is any machine that converts information into different forms using a specific repeatable process. It is widely used in game theory and can be also used in the advancement of computer game design. This paper experiments the design of a standard tic-tac-toe game using Turing machine, a simple abstract computing device that works on a set of predefined rules to determine a result.

Tic-Tac-Toe, also known as noughts and crosses or Xs and Os, is a two-person, paper, and pencil game in which each player alternatively marks a cell in a three-by-three grid with a X or a O. Though the game is not new and is in practice for many years, it involves looking ahead and trying to find out what the opponent decides, in the next step. To win in this game, one needs to make use of a strategy. The winner is the player who successfully places three of their markings in a horizontal, vertical, or diagonal row. A sample three-by-three Tic-Tac-Toe Game grid is presented in Fig. 1. [8-9, 12-14].

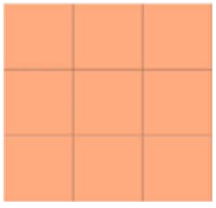

Fig. 1. Tic-Tac-Toe Grid

There are a total of 8 possible winning sequences, three each for horizontal and vertical and 2 for the diagonal as can be seen in Figs. 2, 3, and 4.

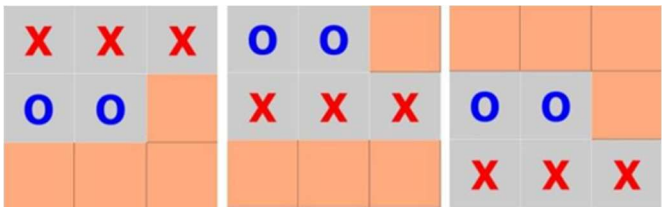

Fig. 2. Horizontal winning sequences

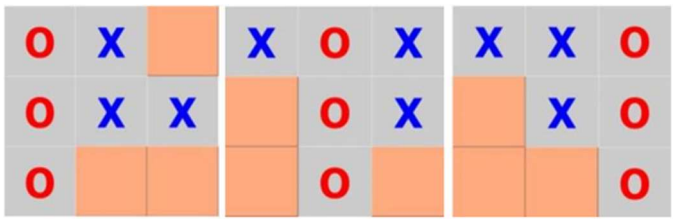

Fig. 3. Vertical winning sequences

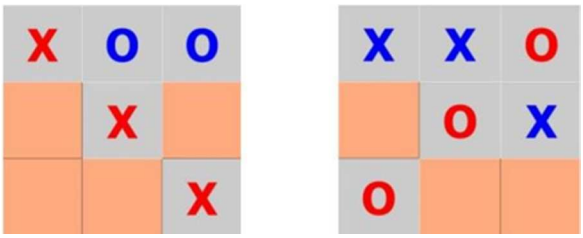

Fig. 4. Diagonal winning sequences

If all the cells are filled and either of the symbols did not form a sequence, then the game is considered as tie, as presented in Fig. 5.

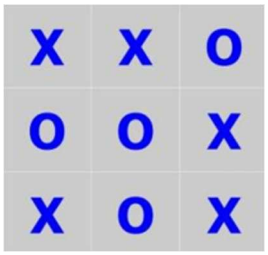

Fig. 5. Tie sequence

In this implementation, each cell or space in the grid is represented by an index as can be seen in Fig. 6. This index position is used to refer to the cell position.

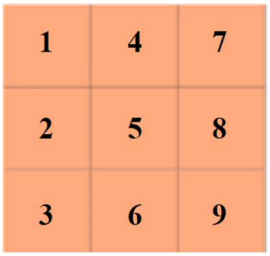

Fig. 6. Indexes are assigned to the spaces

The possible winning sequences represented using indices are {1,2,3}, {4,5,6}, {7,8,9}, {1,4,7}, {2,5,8}, {3,6,9}, {1,5,9}, {3,5,7}.

The paper is organized as follows: Section II describes the related works which is followed by the concepts explaining the multi-tape Turing machine in section III. The proposed approach of the solving od the problem is presented in section IV further to which the paper is concluded in the last section.

II. Related Work

Automata theory plays a major role in many applications that include natural language processing, lexical analysis, and game design. The literature finds its usage in word prediction where a deterministic finite automaton has been used to find the syntactic meaning of the sentence [1-2]. Finite Automata, push down automata and Turing machines are the three types of machines that are generally used to solve any computational problem. Finite automata also called a finite state machine has been used in identifying the concurrent and interleaved actions of human activity recognition research [3]. Finite state machines can also be used to simulate sequential logic or control execution flow.

Turing machine are abstract computational devices that help to find out what can be computed and what cannot. If a Turing machine solves a problem, then the problem is said to be computable. In other terms, if a Turing machine cannot solve a problem, then the problem is not computable in the actual sense. In literature, Turing machines have been used in many fields to solve the problem. This paper uses it to solve the standard Tic-Tac-Toe problem with the help of the Multi-Tape Turing machine, a variation of a standard one. The literature finds the use of Turing machines, in detecting attacks, and errors in the static analysis and gaming as well. This section presents a few of the works in which an automata theory or Turing machine has been used.

The concepts of automata theory, along with the examples, types and variations are discussed in [4-5]. To understand the use of automata in other domains, the authors have designed and discussed a detection and prevention system and analyzed its effectiveness for different injection attacks which lead to the prominent no-SQL databases such as MongoDB [6]. Formal software verification also uses automated analysis in [7] and helps in the detection of errors during the verification process. This paper also presents a short tutorial on three techniques of static analysis with the pros and cons of implementation tools. A ransomware detection system proposed in [8], detects ransomware attacks by assessing the current state of the computer. This detection uses the finite state machine to process the knowledge of ransomware attacks. In [9], an algorithm for model extraction and abstraction is presented which optimizes the size of the extracted model. The properties of extracted models have been compared against declarative specifications to provide the performance measures. This model was implemented in Jive. In [10], the universal Turing machine provides a solution for building and maintaining the recursively enumerable languages which are accepted by Turing machines. This work also experiments with the results using JFLAP. JFLAP is software that helps in experimenting with formal languages and grammar of various automata [11].

In addition to the above-mentioned domains, automata theory has also taken its place in gaming. It also enables the user to convert the automata from one form to another. A foreordained approach discussed in [12] explains a never loose way to play the tic-tac-toe game. Certain axioms and functions have been defined along with a few implications. In [13], tic-tac-toe is being developed using soft computing techniques with the gameplay being programmed using JavaScript, HTML, and CSS. This game demonstrates the game between the player and the computer, and the algorithm helps to decide the next move to win the game or to make it a tie. A winner decision model of the tic-tac-toe game described in [14] uses a Multi-tape Turing machine. But the model implementation holds a higher time complexity if implemented in JFLAP since it has a lot of transitions involved. The work proposed in this paper presents an optimal automaton to implement the tic-tac-toe game in JFLAP using Multi-tape Turing machine. The automata have been modeled in such a way that the automata need not check the unnecessary conditions.

III. Concepts

A. Multi Tape Turing Machines

Multi tape Turing machine has several tapes, each of which will be accessed by a different head as represented in Fig. 7. Every head can move autonomously compared to the remaining heads. In the beginning, the input is presented on

the tape 1 with the assumption that the rest of the tapes are blank. The Turing machine initiates every tape with the assumed symbol and moves to the right till it reaches the end of the input. The transition function of the Turing machine is given as:

$$\delta: Q \times \Gamma^n \rightarrow Q \times \Gamma^n \times \{L, R, S\}^n$$

Here n is number of tapes. An example function will look like:

$$\delta(q_i, a_1, \ldots, a_n) = (q_j, b_1, \ldots, b_n, L, R, \ldots L)$$

This implies that the machine is in state q_i, on reading a_1 through a_n, it goes to the state q_j and changes the symbols to b_1 through to b_n and moves its head to the right or left or it stays as it is based on the symbol used.

Although Multi-tape Turing machines appear to be more powerful than standard Turing machines, we can demonstrate that their power is similar. We know that two machines are equivalent in power if they can recognize the same language. Each Multi-tape Turing machine possesses a single tape Turing machine. But at the same time, when the discussion is on the speed of recognition of a language, for a few examples, one should accept that the Multi-tape Turing machines is faster than the standard Turing machine [4, 5].

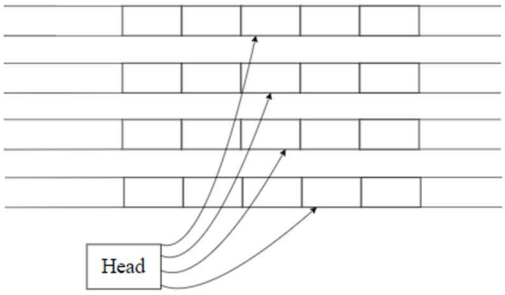

Fig. 7. Multi-tape Turing Machines

IV. IMPLEMENTATION

A. Algorithm:

This section presents the implementation of the tic-tac-toe game using Multi-tape Turing machine M. The outline of the algorithm is presented in Fig. 8.

Input: A complete sequence of the X's and O's in alternate places each representing a different player.
Output: Declaration of the winner.

This proposed implementation uses 4 tapes for the game. The details of the usage of the tapes are described below:

- 1st- tape: To take the input sequence i.e., the indexes sequence in which players place the symbols X and O.
- 2nd- tape: X player's sequence will be stored in this tape. From the 1st tape, the indexes sequence of X will be stored.
- 3rd- tape: The O player's sequence will be stored in this tape. From the 1st tape, the indexes of a sequence of O will be stored.
- 4th-tape: The winning sequences will be stored in this tape and used to check whether X or O won the game or it's a TIE.

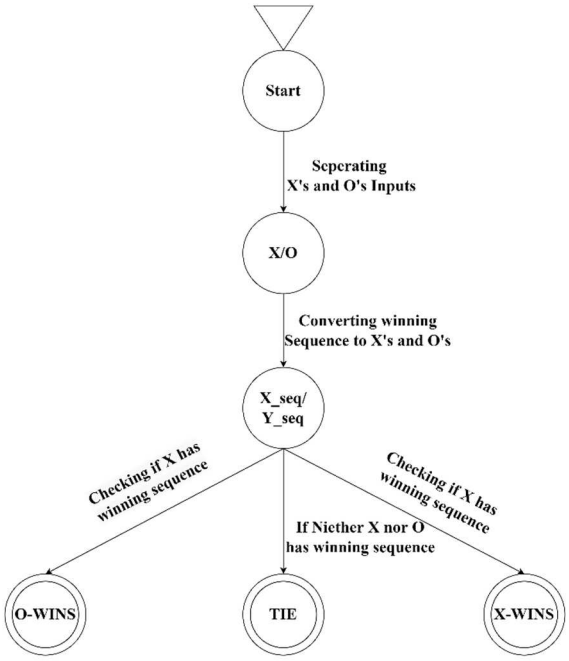

Fig. 8. Workflow of Algorithm

Step1: At the X/O node, the input sequence will be divided into X-player's and O-player's sequences which are stored in the 2nd tape and 3rd tape respectively.

Step2: At the X_seq/O_seq node, it converts the winning sequence in the 4th tape to the possible sequences of both the X-player's and O-players. It first checks the winning sequence for the X, for each number (index) in the X sequence, it traverses through the 4th tape and if the sequence in the 2nd tape comes across the same sequence in the 4th tape, then it changes to X.

Step3: At the X_seq/O_seq node, after converting the winning sequence of the X. It starts checking the winning sequence for the O, for each number (index) in the O sequence, it traverses through the 4th tape and if the sequence in the 2nd tape comes across the same sequence in the 4th tape, then it changes to O.

Step4: From X_seq/ O_seq node, it checks for the winning sequence from tape-4. In tape-4, if it finds 3 consecutive X's it reaches to final X-win node and gives the conclusion as X player wins, on the other hand, if it finds 3 consecutive O's it reaches to O-win node and gives the conclusion as O player wins. If it did not find the consecutive sequences of X's and O's after traversing the entire 4th tape, it reaches to TIE node and gives the conclusion as the game is TIE.

B. Automata design:

The detailed transition functions are presented here which will help in implementing the described problem.

δ (Start, P, P, P, P)=δ (X, B, B, B, P, R, R, R, S)

δ (X, {1-9},b, b, P) =δ (O, B, {1-9}, b, P, R, R, S, S)

δ (O,{1-9},b ,b ,P) = δ (X, B, b, {1-9}, P, R, S, R, S)

δ (O, B, b, b ,P) = δ (X_seq, B ,E ,E ,P ,S ,L ,S ,R)

δ (X, B, b, b ,P) = δ (X_seq, B ,E ,E ,P ,S ,L ,S ,R)

δ (X_seq, B,{1-9},E,{1-9,X ,W}) = δ (X_seq, B, {1-9}, E, {1-9,X,W},S,S,S,R)

δ (X_seq, B,{1-9}, E, E) = δ (X_T, B,{1-9},E,E,S,S,S,L)
δ (X_T, B, b, E,{1-9,X,W}) = δ (X_T, B, b, E,{1-9,X,W},S,S,S,L)

δ (X_T, B, b, E, P) = δ (X_seq, B, b, E, P, S, L, S, R)

δ (X_seq, B, B, E, {1-9, X}) = δ (O_seq, B, B, E, {1-9,X}, S, R, L, S)

δ (O_seq, B, b, {1-9}, {1-9, X,O,W}) = δ (O_seq, B, b, {1-9}, {1-9,X,O,W}, S, S, S, R)

δ (O_seq, B, b, {1-9}, E) = δ (O_T, B, b, {1-9}, E, S, S, S, L)

δ (O_T, B, b, b,{1-9, X , O, W}) = δ (O_T, B, b, b,{1-9, X, O, W}, S, S, S, L)

δ (O_T, B, b, b, P) = δ (O_seq, B, b, b, P, S, S, L, R)

δ (O_seq, B, b, B,{X,O}) = δ (W_C, B, b, B,{X,O}, S, S, R, S)

δ (W_C, B,b,b,E) = δ (Draw, B, b, b, E, S, S, S, S)

δ (W_C, B, b, b, O)= δ(W_O_1,B,b,b,O,S,S,S,R)

δ (W_O_1, B, b, b, O) = δ (W_O_2, B, b, b, O, S, S, S, R)

δ (W_O2, B, b, b, O) = δ (O_win, B, b, b, O, S, S, S, R)

δ (W_C, B, b, b, X) = δ (W_X_1, B, b, b, X, S, S, S, R)

δ (W_X_1, B, b, b, X) = δ (W_X_2, B, b, b, X, S, S, S, R)

δ (W_X_2, B, b, b, X) = δ (X_win, B, b, b, X, S, S, S, R)

δ (W_O_1, B, b, b,{1-9, W, X}) = δ (W_C, B, b, b,{1-9, W, X}, S, S, S, R)

δ (W_O_2, B, b, b,{1-9,X}) = δ (W_C, B, b, b,{1-9,X},S,S,S,R)

δ (W_X_1, B, b, b,{1-9,W,O}) = δ (W_C, B, b, b,{1-9, W, O},S, S, S, R)

δ (W_X_2, B, b, b,{1-9, O}) = δ (W_C, B, b, b,{1-9, O}, S, S, S, R)

where Start is the initial state, Draw, O_win, and X_win are the final states.

C. Input format:

Fig. 9. Input format

To experiment with the proposed approach, the Turing machine is designed using JFLAP. The snapshots showcasing the various scenarios of JFLAP are presented in Figs. 9 through 16. Every tape's input starts with a 'P', the first tape should contain the sequence of X's and O's placements one after the other and ends with 'E', the second tape contains b's where the number of b's is one more than the total number of X's placements, the third tape contains b's where the number of b's is one exceeding than the total number of O's placements and the fourth tape contains the winning sequences, separated with a 'W' after each of them and after the last sequence it contains an 'E'.

Fig. 10. Initially when the input is loaded

Fig. 11. Separating X's and O's inputs

Fig. 12. Converting the winning sequence to X's

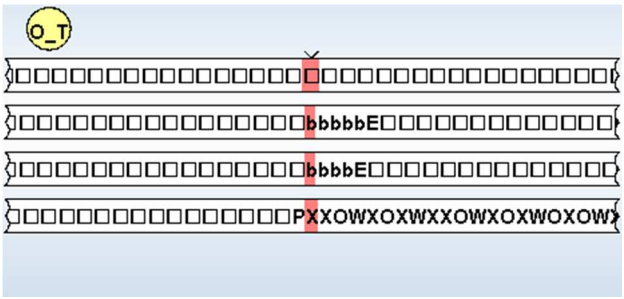

Fig. 13. Converting the winning sequence to O's

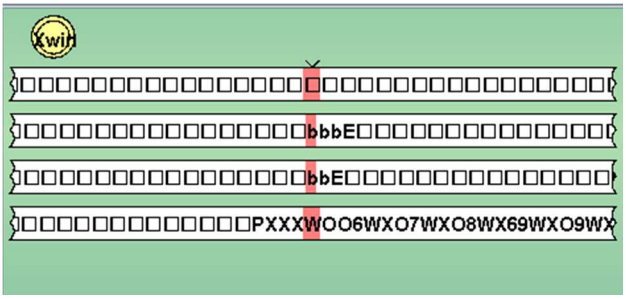

Fig. 14. JFLAP output if X wins

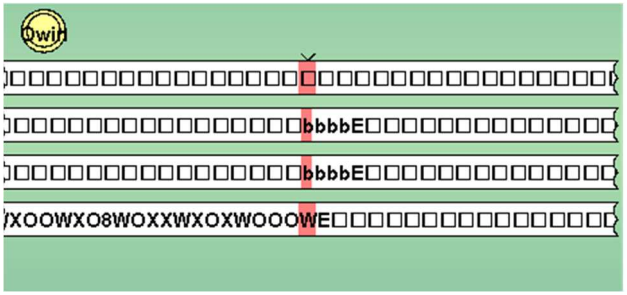

Fig. 15. JFLAP output if O wins

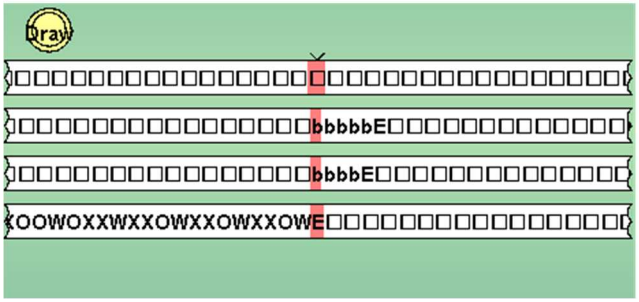

Fig. 16. JFLAP output if Tie

This paper has experimented with the use of finite state machines in game design. Though the literature presents the use of state machines in various domains like string matching, plagiarism detection, word extraction, etc., the implementation in game design simplifies the complex behaviors into a finite set of states. Especially Turing machines which are considered to be one of the most powerful machines in the automata theory can take the implementation of games to a new dimension.

V. Conclusion

This paper proposes an optimized version of the sue of Multi-tape Turing machine to implement the Tic-Tac-Toe game. The proposal uses 4 tapes and experiments the same in JFLAP. It is a decision-making model which declares whether X wins or O wins or a tie. It is an optimal implementation of the tic-tac-toe game using automata theory while compared to the existing work since it considers optimal assumptions such as starting the game with X by default which decreases the compexity of the model by avoiding unnecessary transitions.Further, the JFLAP Source code can be used, and an actual tic tac toe game can be developed using the same model.

REFERENCES

[1] Kanjirangat, Vani & Gupta, Deepa. (2018). Unmasking text plagiarism using syntactic-semantic based natural language processing techniques: Comparisons, analysis and challenges.

[2] Tamizharasan, M., Shahana, R.S., Subathra, P."Topic modeling-based approach for word prediction using automata",Journal of Critical Reviews,PP 744-749,2020.

[3] Kavya, J., & Geetha, M. (2016). An FSM based methodology for interleaved and concurrent activity recognition. Paper presented at the 2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016, 994-999.

[4] Anil Maheshwari and Michiel Smid, School of Computer Science, Carleton University, Canada, "Introduction to Theory of Computation", pp. 137-150.

[5] Michael Sipser, Massachusetts Institute of Technology , "Introduction to Theory of Computation - second Edition", pp. 148-149.

[6] Swathy Joseph and K.P. Jeevitha, An Automata Based Approach for the Prevention of NoSQL Injections, In security in Computing and Communications(link is external), Springer – Communications in Computer and Information Science(link is external), 2015, pp 538-546. Print ISBN 978-3-319-22914-0, Online ISBN 978-3-319-22915-7.

[7] V. D'Silva, D. Kroening and G. Weissenbacher, "A Survey of Automated Techniques for Formal Software Verification," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 7, pp. 1165-1178, July 2008, doi: 10.1109/TCAD.2008.923410.

[8] Ramesh, G., Menen, A. “Automated dynamic approach for detecting ransomware using finite-state machine”, Decision Support Systems.2020.

[9] K. P. Jevitha, Swaminathan J., Jayaraman, B., and M, S., “Finite-state model extraction and visualization from Java program execution”, Software: Practice and Experience, vol. 51, pp. 409-437, 2021.

[10] Pradhan, Tribikram. (2015). Enhancement of Turing Machine to Universal Turing Machine to Halt for Recursive Enumerable Language and its JFLAP Simulation. International Journal of Hybrid Information Technology. 8. 193-202. 10.14257/ijhit.2015.8.1.17.

[11] Susan Rodger and Thomas Finley, JFLAP - An Interactive Formal Languages and Automata Package, ISBN 0763738344, Jones and Bartlett, 2006.

[12] A. Singh, K. Deep and A. Nagar, "A "Never-Loose" Strategy to Play the Game of Tic-Tac-Toe," 2014 International Conference on Soft Computing and Machine Intelligence, 2014, pp. 1-5, doi: 10.1109/ISCMI.2014.13.

[13] S. Karamchandani, P. Gandhi, O. Pawar and S. Pawaskar, "A simple algorithm for designing an artificial intelligence based Tic Tac Toe game," 2015 International Conference on Pervasive Computing (ICPC), 2015, pp. 1-4, doi: 10.1109/PERVASIVE.2015.7087182.

[14] S. Garg and D. Songara, "The winner decision model of tic tac toe game by using multi-tape turing machine," 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2016, pp. 573-579, doi: 10.1109/ICACCI.2016.7732107.