International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 12 | December - 2022 Impact Factor: 7.185 ISSN: 2582-3930

Simulation of Pushdown Automata using Python

Aditya Akangire
AI and DataScienceVIT
Pune Pune, India

Abdul Mueed
AI and DataScienceVIT
Pune Pune, India

Kartik Rupauliha
AI and Data Science
VIT Pune Pune, India

Ayush Vispute
AI and DataScienceVIT
Pune Pune, India

Sarthak Akkarbote
AI and Data Science
VIT Pune
Pune, India

***Abstract*— In the same approach that we create DFA for regular grammar, a pushdown automaton (PDA) is a technique for implementation of context- free grammar. Because of an extra memory segment called stack, a PDA can remember an immense amount of information, whereas a DFA can only remember a certain amount of information. In our project, we have implemented PDA using Python, and made simulations of four PDAs. To check if the input string has n number of 0s followed by n number of 1s, palindrome, if no. of 0s and 1s are equal or not and last one to check if, number of 1s are twice as compared to number of 0s. We also print transition tables for better understanding of PDA transitions and stack operations.**

Keywords—python,pushdown automata, palindrome,automata theory, stack.

INTRODUCTION

Pushdown Automata is a type of finite automaton which includes an additional memory known as a stack that allows it to recognise Languages that are free of context. A Stack is a linear type of data structure where all insertions and deletions are made only from one end, the top. We can perform two basic operations on stack, insertion of an element known as 'push' and deletion of an element known as 'pop' strictly from the top only. Pushdown Automata(PDA) is one of the extensively important and widely used topics in Theory of Computation(TOC). It has a huge number of applications and therefore, we decided to design some examples of Pushdown Automata and implement it withthe help of Python programming.

Formal Definition of PDA can be given as :

- Q : Non empty finite set of states
- ∑ : Input alphabets
- Γ : Stack alphabets
- q0: Initial state
- Z : Stack start symbol(bottom of stack)
- F : the final state, δ is a transition feature which maps Q x {Σ ∪∈} x Γinto Q x Γ*. In each state, PDA will read symbols from top of the stack and shift to a new state and change the symbol of stack.
- Components of Pushdown Automata :

Input tape : The input tape is subdivided into cells or symbols. The input head is read-only i.e it can not change the symbol and can only move one symbol from left to right at a time.

Finite control : It has a pointer which points to the currently being read symbol.

Stack : This is a kind of data structure in which data could only be put or removed from one end. It's size is unlimited.. It is used to temporarily store items in a PDA.

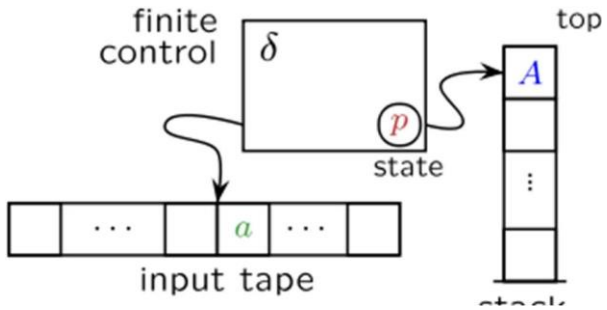

Fig 1

Image ref : https://en.wikipedia.org

According to A state-transition table is used in automata and sequential logic to show which state (or states in NFA) a finite-state machine will transition to given the present situation and other inputs.

Considering the fact that Pushdown automata have an

extra memory segment called stack which makes it possible for PDA to remember infinite amounts of information, unlike finite state automata which can store only finite amounts of information. The paper covers certain languages which can be designed using pushdown automata. The programs are written in Python and have used Python's library named automata.

I. LITERATURE SURVEY

A. Non-Deterministic PDA Tool[1]

Similar to our project, this tool is being used for creating simulations of PDA. The program is in Java language. The user is able to run PDA with many inputs at around the same moment. User has to specify all necessary information in an input file. This tool is able to stimulate PDAs in the form of graphs so that it could assist the user to follow the steps as per simulation.

B. Pushdown Automata Simulator[2]

This is a GUI based java program which has created a simulator to study the behavior of PDA in the form of visualization. This type of simulator can handle both vacant stack as well as the end state techniques of approval. It allows users to easily transform the PDA which takes a vacant stack into a PDA which accepts by end state. It is a similar process the other way around.(Hamada 1)

C. Autonomous push-down automaton built on DNA*[5]

In this paper, the authors have presented a conceptual model on execution of pushdown automata built on DNA. They were persuaded by The Turing Machine model and the finite automaton model. Furthermore, they have discussed the basic introduction of the pushdown automaton being a non-deterministic finite automata and also a schema of the PDA. A few solved examples of PDA are also stated for better understanding of the concept. Then, it has been mentioned about the basic elements of the implementation of The PDA with respect to a DNA molecule. Further, they've discussed the empirical implementation and the transition rules along with their molecular representation. The paper concludes with successfully presenting a contemporary way to implement PDA contingent to DNA molecules along with restriction enzymes. (KRASIŃSKI et al. 1)

D. Pushdown Automata in Statistical Machine Translation[6]

This paper contains the application of PDA in relation with statistical machine translation. They've also mentioned SCFGs, HiPDT and described decoding in three steps: Translation, Language Model Application and Search. They've researched in such a way proving that PDAs are more worthy in decoding SCFGs and other language models. (Allauzen et al. 1)

E. Pushdown Automata[7]

This paper contains the theory and information about Pushdown Automata and then further discusses the similarities between Pushdown Automata and context free grammar and then they further talk about various properties and types of Pushdown Automata. The paper then discusses how recursion is the face and the building block of Pushdown Automata. The paper finally discusses some of the machine models based on Pushdown Automaton.

II. PROBLEM STATEMENT

1. We will be designing a PDA system for accepting the language in the form 0^n1^n and 0^n1^{2n}. The important thing to note is, the order of 0's and 1's should be maintained.

2. L={x∈{o,1}*/#o=#1} where # represent no. of zero i.e. 0's = 1's.

3. We will be designing a Python implementation of palindrome numbers using Pushdown Automata with the alphabets 0 and 1.

III. METHODOLOGY

The python program to simulate PDA consists of two .py files and for text files of four different PDAs simulated.

First python file is FileHandler.py in which two functions namely readFile() and parseFile() are defined. readFile() reads the input file and displays an error message if the proper input file is not provided. parseFile() function is used to assign a variable to each line from text files and returns a dictionary containing these variable and value(text) pairs.

PDA.py is the second python file which performs the main operations of the program.The compute() function does the stack operations and generates transition table. main() function asks for user input and displays relevant messages accordingly.

text files contain following details to define a PDA:Total States on Line 1

Input Word Symbols(0,1) on Line 2Stack Symbols 3 Initial State Symbol on Line 4 Initial Stack Symbol on Line 5List of Final States on Line 6

Productions in the form of (Current State, Current Input Symbol, Current Top of Stack, Next State, Push/Pop Operation Symbol) from Line 7 and onwards.

Following PDAs are implemented in this program

1) $0^n 1^n$

1. So, initially a special symbol "$" is being put in the empty stack.
2. Then, checking in the input string, if we encounter "0" and at the top of the stack lies "$", we push "0" intothe stack.
3. If the next letter we encounter is "1", and at the top of the stack lies "0", then we pop the "0" out of the stack.
4. So, overall we push the "0"s into the stack and pop out one "0" for every "1" we get in the input string.
5. Finally, after encountering all the letters of the input string, we found the "$" symbol in the stack, then the string is being accepted otherwise not.
6. We also check that the input string is strictly of the form $0^n 1^n$, $n \geq 1$.

2) WCW^R (Palindrome)

1. Here, in an ideal case "W" is a combination of one string, "C" acts as a center/separator of the input stringand "W^R" is another string opposite of "W".
2. So, initially a special symbol "$" is being put in theempty stack.
3. Then, all the alphabets of string "W" are being pushed into the stack.
4. The next alphabet encountered will be "C" which weare going to ignore.
5. The next set of strings will be of "W^R", as the encountered alphabets start matching with the alphabets at the top of the stack, we'll start popping them out.
6. Finally, after encountering all the alphabets, if we find "$" symbol in the stack, then the string will be accepted otherwise not.
7. We also check whether or not the input string is of the form WCW^R.

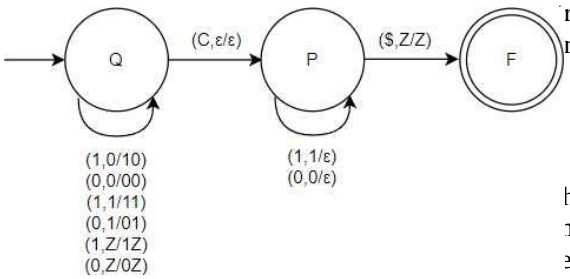

Fig. 2

3) $0^n 1^{2n}$

1. So, initially a special symbol "$" is being put in theempty stack.
2. Then, checking in the input string, if we come across "0" and "$" is present as the up most elementof the stack, we push "0" into the stack.
3. Again if we encounter "0" and the up most elementof the stack is "0", we push "0" into the stack.
4. When we encounter "1" and the up most element of the stack is "0", we move to q_1 state from initial q_o state.
5. If we again encounter "1" in q_1 state and the up most element of the stack is "0", we pop a "0" out ofthe stack and move to q_2 state.
6. Again if we encounter "1", we move from q_2 to q_1 state and then for immediate next "1" we as above(5).
7. Finally, after encountering all the letters of the input string, we found the "$" symbol in the stack, then the string is accepted otherwise not.

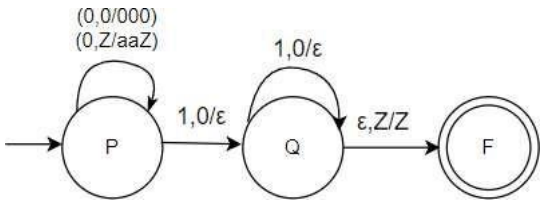

Fig. 3

4) Number of 0s = Number of 1s

1. Initially a special symbol "$" is being put in the empty stack.
2. Then, checking in the input string, we encounter "0" or "1" and we push it into the stack. That means if "0" comes first, push it in the stack.
3. After "0" if again "0" comes then push it in the stack.
4. If "1" comes first, push it in the stack ("0" did not come yet).If again "1" comes then push it in the stack.
5. Now if "0" lies on top of stack and we encounter "1" then, pop "0"'

Similarly if "1" is present on top of the stack and we encounter "0" then, pop "1".

6. Finally, after encountering all the letters of the input string, we found the "$" symbol in the stack, then the string is being accepted otherwise not.

IV. RESULT

This project covers four examples of the implementation of pushdown automata. The results will be shown by a string which will be either "The string is accepted by PDA" or "The string is not accepted by PDA". The user could select one out of the four options which are $0^n 1^n$, WCW^R, $0^n 1^{2n}$ and No. of 0's equals to No. of 1's. After running the program and inputting the string, the user will get to know whether or not the string is accepted by The Pushdown Automata.

V. FUTURE SCOPE

This model helps us to easily understand PDAs and study them properly with the help of transition tables. In the future we can design and add more PDAs in the format of text files and implement it using the same algorithm.

VI. SCOPE OF PROJECT

There is an immense scope for this project which can be worked upon. Since only a few simulations have been performed, a greater number of simulations can be done using Python. Other languages other than Python may also be used for simulation. The simulation can also be shown graphically step by step and then maybe the comparison between different languages can be done depicting which language takes a minimum amount oftime.

VII. CONCLUSION

We have considered the problems of push down automata.These problems can be seen as model checking and context-free properties for pushdown models.We have used Python for checking PDA for Palindrome, if number of 0s and 1s are equal or not and if no of 1s are twice as compared to number of 0s.We have seen how the system works and what are the methods of Data Collection and Analysis.

ACKNOWLEDGEMENT

We are grateful to our project guide for shaping our ideas and constant support during the project. Such projects help us a lot to develop skillful and innovative thinking. Thanks to The Department of IT and AI & Data Science, Vishwakarma Institute of Technology for including it in our curriculum. Special acknowledgement to all our group members for such efforts and exchange of various concepts and ideas which made the project possible. Last but not the least, we are thankful to our parents and friends for their support and encouragement.

REFERENCES

[1] Another non-deterministic push-downautomaton. available at http: //www.cs.binghamton.edu/˜software/pda/pdadoc.html.

[2] Pushdown Automata Simulator by FelixErlacher

[3] https://research.cs.queensu.ca/home/ksalomaa/julk/p47-okhotin.pdf

[4] https://www.geeksforgeeks.org/introduction-of-pushdown-automata/

[5] https://www.researchgate.net/publication/51959108_Autonomous_push-down_automaton_built_on_DNA

[6] https://aclanthology.org/J14-3008.pdf

[7] https://www.researchgate.net/publication/242427937_Pushdown_Automata/link/0a85e532de95307b1e000000/download

[8] https://www.tutorialspoint.com/automata_theory/pushdown_automata_introduction.htm

[9] https://en.wikipedia.org/wiki/State-transition_table

[10] Pushdown Automata Hendrik Jan Hoogeboom and Joost Engelfriet

[11] Head Pushdown Automata Samson Ayodeji Awe

[12] RE-DESIGNING THE PACMAN GAME USING PUSH DOWN AUTOMATA

[13] Complexity of Input-Driven Pushdown Automata1 Alexander Okhotin2 Kai Salomaa3

[14] Atig, M. F., Bollig, B., & Habermehl, P. (2017). Emptiness of Ordered Multi-Pushdown Automata is 2ETIME-Complete. *International Journal of Foundations of Computer Science*, *28*(08), 945–975. https://doi.org/10.1142/s0129054117500332

[15] Fransson, T. (2013). *Simulators for formal languages, automata and theory of computation with focus on JFLAP* [Student thesis, Mälardalens högskola, Akademin för innovation, design och teknik]. http://urn.kb.se/resolve?urn=urn:nbn:se:mdh:diva-18351

[16] LIN, H. J., & WANG, P. S. P. (1989). PUSHDOWN RECOGNIZERS FOR ARRAY PATTERN. *International Journal of Pattern Recognition and Artificial Intelligence*, *03*(03n04), 377–392. https://doi.org/10.1142/s0218001489000292

[17] Vayadande, Kuldeep, Ritesh Pokarne, Mahalaxmi Phaldesai, Tanushri Bhuruk, Tanmai Patil, and Prachi Kumar. "SIMULATION OF CONWAY'S GAME OF LIFE USING CELLULAR AUTOMATA." *International Research Journal of Engineering and Technology (IRJET)* 9, no. 01 (2022): 2395-0056.

[18] Vayadande Kuldeep, Ram Mandhana, Kaustubh Paralkar, Dhananjay Pawal, Siddhant Deshpande, and Vishal Sonkusale. "Pattern Matching in File System." *International Journal of Computer Applications* 975: 8887.

[19] Vayadande Kuldeep, Neha Bhavar, Sayee Chauhan, Sushrut Kulkarni, Abhijit Thorat, and Yash

Annapure. *Spell Checker*

Model for String Comparisonin Automata. No. 7375. EasyChair, 2022.

[20] VAYADANDE KULDEEP. "Simulating Derivations of Context-Free Grammar." (2022).

[21] Vayadande, Kuldeep, Neha Bhavar, Sayee Chauhan, Sushrut Kulkarni, Abhijit Thorat, and Yash Annapure. *Spell Checker Model for String Comparison in Automata*. No. 7375. EasaafyChair, 2022.

[22] Varad Ingale, Kuldeep Vayadande, Vivek Verma, Abhishek Yeole, Sahil Zawar, Zoya Jamadar. Lexical analyzer using DFA, International Journal of Advance Research, Ideas and Innovations in Technology, www.IJARIIT.com.

[23] Kuldeep Vayadande, Harshwardhan More,Omkar More, Shubham Mulay,Atahrv Pathak, Vishwam Talanikar, "Pac Man: Game Development using PDA and OOP", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072, Volume: 09 Issue: 01 | Jan 2022, www.irjet.net

[24] Kuldeep B. Vayadande, Parth Sheth, Arvind Shelke, Vaishnavi Patil, Srushti Shevate, Chinmayee Sawakare, "Simulation and Testing of Deterministic Finite Automata Machine," *International Journal of Computer Sciences and Engineering*, Vol.10, Issue.1, pp.13-17, 2022.

[25] Rohit Gurav, Sakshi Suryawanshi,Parth Narkhede,Sankalp Patil,Sejal Hukare,Kuldeep Vayadande," Universal Turing machine simulator", International Journal of Advance Research, Ideas and Innovations in Technology, ISSN: 2454-132X, (Volume 8, Issue 1 - V8I1-1268, https://www.ijariit.com/

[26] Kuldeep Vayadande, Krisha Patel, Nikita Punde, Shreyash Patil, Srushti Nikam, Sudhanshu Pathrabe, "Non-Deterministic Finite Automata to Deterministic Finite Automata Conversion by Subset Construction Method using Python," *International Journal of Computer Sciences and Engineering*, Vol.10, Issue.1, pp.1-5, 2022.

[27] Kuldeep Vayadande and Samruddhi Pate and Naman Agarwal and Dnyaneshwari Navale and Akhilesh Nawale and Piyush Parakh," Modulo Calculator Using Tkinter Library", EasyChair Preprint no. 7578, EasyChair, 2022