A Review Paper on Pushdown Automata Implementation

Aditya Akangire, Sarthak Akkarbote, Kartik Rupauliha, Ayush Vispute and Abdul Mueed

March 29, 2022

A Review Paper on Pushdown Automata Implementation

Aditya Akangire
AI and Data Science
VIT Pune
Pune, India

Sarthak Akkarbote
AI and Data Science
VIT Pune
Pune, India

Kartik Rupauliha
AI and Data Science
VIT Pune
Pune, India

Ayush Vispute
AI and Data Science
VIT Pune
Pune, India

Abdul Mueed
AI and Data Science
VIT Pune
Pune, India

***Abstract*— Pushdown Automata is a finite automaton with an additional data input segment called stack that helps in recognizing Context Free Languages. We can compare it to finite automata, but the exception is that because of stacks, it is able to handle infinite strings. This paper consists of review and survey of other projects related to Pushdown Automata implementation and Comparative study of other projects which have implemented PDA. Theoretical information of PDA would help to understand it better and design our problem statement. We plan to design PDA applications that will help to study PDA operations properly with the help of transition tables.**

Keywords—pushdown automata, palindrome, automata theory

Introduction

A push down automaton (PDA) as shown on[4] is a procedure for creating a grammar without context in the same way that a regular grammar is implemented using DFA. A DFA is able to remember or store only a certain amount of data, but a PDA has the ability which allows it to keep unlimited data because it has an additional memory section known as the stack.
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. [5]
Formal Definition of pushdown automata can be defined as :

- Let S be the states
- Let ∑ be the input symbols
- Γ is the set of pushdown symbols
- Let P be the PD symbol
- Let F be the last state δ is a transition feature which maps S x {Σ ∪∈} x Γinto S 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.
- A PDA is more capable than an FA. Any language that is acceptable to FA is likewise acceptable to the PDA. PDA is able to accept a class of languages that FA can not accept. As a result, PDA is very much superior to FA.
- Components of Pushdown Automata[4] :

Input tape : The input tape is separated into several cells or symbols. The input head is read-only and can only move one symbol at a time from left to right.
Finite control : The finite control has a pointer that points to the currently being read symbol.
Stack : The stack is a structure in which objects may only be pushed and removed from one end. It is unlimited in size. The stack is used to temporarily store items in a PDA.

According to [6] In automata as well as sequential logic, there is a state-transition table which displays which state (or states in NFA) a finite-state machine will transition to considering the current situation as well as other inputs. [6]This is a form of truth table that accommodate the present state as well as other inputs as inputs and the next state as well as other outputs as outputs.

1. Literature Survey

A. Non-Deterministic PDA Tool[1]

It's a tool used for simulating PDA which is similar to our project. 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 another simple GUI based java program which has developed a simulator to study the behavior of PDA by 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.

2. Problem statement

1. We will be designing a PDA system for accepting the language in the form 0^n1^n. Here,we need to maintain the order of 0's and 1's.
2. L={x∈{o,1}*/#o=#1} where # represent no. of zero i.e. 0s in x = 1s in x.
3. L = {wcv | w={0, 1}*}

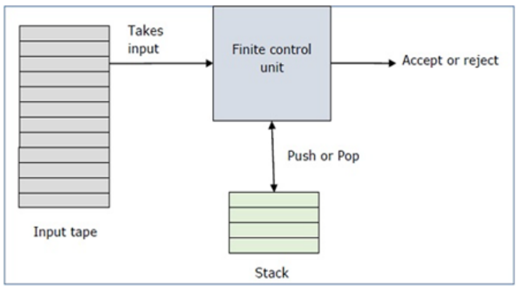
3. Proposed system

Our project is such that the user is first asked to input the file path. He/she can choose among the following file paths:
1. 0n1n: If the user inputs this path, he/she will then be asked to input a string. This language accepts L = .0^n1^n, n≥1.
2. 0n-1-2n: If the user inputs this path, he/she will then be asked to input a string. This language accepts L = 0^n1^{2n}, n≥1.
3. n0n1: If the user inputs this path, he/she will then be asked to input a string. This path is to check whether the 0s in string are in the same number as 1s in string.
4. wcwR: If the user inputs this path, he/she will then be asked to input a string. This path is to check whether the input string is a palindrome or not.

After inputting the string, the push-pop operations will commence in the stack memory. If, after all the operations the stack memory is empty, the string will be accepted in the PDA.
Finally, if the STACK is empty, a message will appear "String is accepted by PDA" else the message will occur "String is rejected by PDA".

4. Data collection method and analysis

There is an extra memory section called stack in Push Down Automata (PDA) that aids in the recognition of context-free language.



The input tape collects data and sends it to the Finite Control Unit, which determines whether to push the data into the stack, move it to another state, or pop something from the stack. When all the data is obtained, the final state and stack are checked, and the string is either accepted or not accepted depending upon the results.

5. Comparison with existing systems

There exist multiple machine models that have data types that are similar to the topic in discussion which is the pushdown stack. There have been various similar machine models that existed prior to this and a few that are still being used. These machine models are alternates or types of PDA and have been mentioned below.

Simple grammars:

Simple form is when it is in Greibach normal form, which is a special case in which we do not observe 2 productions. They directly correlate to PDAs that are solitary, immutable, or real-time. Although, in some cases, we can remove the instantaneous PDAs.

Two stacks:

Turing power is very well prevalent in finite state appliances which have two stacks. The two stacks in a combination are used as an operating tape, and indeed the 1 PDA 19 equipment could really start moving both paths upon the tape, going to pop and trying to push the components of one stack to another.

Counter automaton:

In this type of model the stack is to string of a fixed symbol * another symbol is restricted. We end up with a natural no. upon which we can perform operations like incrementing, decrementing, and testing for zero. An automaton of this sort does have a class name which contains a numeric value (Z) that could be slightly increased, slightly decreased, or evaluated for nil. Automata with both such similar counters are able to code and perform operations on strings; this makes them very similar to Turing machines.

Blind/Partially blind type of counters:

Let there be a machine that can assume values that are both positive and negative and this machine is able to set all counters to zero in its final state. The machine is blind in the case that the exertion hangs on the state and input only but not on the way the counter is configured. They get termed as partially blind in the case where it blocks at the time the counter is in negative form but does not consider the possibility of any of the counters being 0. Partially blind multi-counters can be very well used for simulating a very complex system of Petri nets.

Finite turn PushDown Automaton:

A finite turn is a PushDown Automaton where the amount of times this same procedure gets toggles between pushing and popping is pre-established. As there is considered to be a constraint of having only a single turn, In contrast to finite turn PDA, which produces ultralinear context-free grammars, this results in linear languages.

Alternation:

A certain non-deterministic automaton is deemed greatly successful when it has a calculation which interprets the information and arrives at an inclusive arrangement then it is considered to be successful. In the case of this type of automata, the starting steps involved in a certain arrangement cause acceptance and are added. So, we can conclude that states and configurations can be non-deterministic or can also be universal. Hence, we have a dual mode system. It is important to note that just standard languages can be accepted by interchanging finite automata.

Two-way PDAs:

The entry path is a double-way mechanism, so we call it a two-way PDA because of its nature. A key step in the process is labeling both ends of the tape that accepts the entry, and thus the double-way PDA can pinpoint the entry's ending and beginning points. Such devices can scan their feed two times as well as in backwards order. This allows for the recognition of non-contextual languages. Eventually, we have multi-head PDAs which are either deterministic or non-deterministic (!), and indeed the class P of languages are recognizable in deterministic polynomial time. The deterministic (k 1)-iterated exponential time complexity classes are better described using multi-headed k-iterated PDAs.

Stack type automata:

This form of PDA has the extra capacity to analyze its stack. In read-only mode it may vary its position which is up and down the stack, thus, its contents exist without change.Stack automata have been shown to be equivalent to PDAs when they do not interpret input while the testing procedure is in progress of the given stack, they are said to be equivalent to PDAs. A recursive stack automaton could indeed begin a fresh stack in between cells of the existing stack. However, before the machine could perhaps advance in the authentic stack, this latest stack must be entirely erased. These automata are analogous to push down-of-PDA, which can also be referred to as ordered grammatical structures.

6. FUTURE SCOPE

The extensive review of Push down automata had also revealed a variety of intriguing different theories and models with unsettled complex issues.One is able to study the various sophisticated applications of pushdown automata to a great extent and consider the complexity of pushdown automata(PDA) in various fields of operation. Currently there have been four examples that have been worked on but upon further investment of time and effort, more can be worked out. Numerous simulations can be made using this model.

7. 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.

REFERENCES

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

[2] Pushdown Automata Simulator by Felix Erlacher

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

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

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

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

[7] Pushdown Automata Hendrik Jan Hoogeboom and Joost Engelfriet

[8] Head Pushdown Automata Samson Ayodeji Awe

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

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

[11] 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

[12] 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

[13] 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

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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

[21] 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.

[22] 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/

[23] 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.

[24] 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