A Review Paper on CFG To PDA Converter

Sanika Kadam  
*Artificial Intelligence and Data Science*  
*Vishwakarma Institute of Technology,*Pune,411037, Maharashtra.  
sanika.kadam20@vit.edu

Aditya Kanawade  
*Artificial Intelligence and Data Science*  
*Vishwakarma Institute of Technology,*Pune,411037, Maharashtra.  
aditya.kanawade20@vit.eduOmkar Kale  
*Artificial Intelligence and Data Science*  
*Vishwakarma Institute of Technology,*Pune,411037, Maharashtra.  
omkar.kale20@vit.edu

Sejal Kharche  
*Artificial Intelligence and Data Science*  
*Vishwakarma Institute of Technology,*Pune,411037, Maharashtra.  
sejal.kharche201@vit.eduAnish Kamble  
*Artificial Intelligence and Data Science*  
*Vishwakarma Institute of Technology,*Pune,411037, Maharashtra.  
anish.kamble20@vit.edu

*Abstract*— Grammar Induction (or Grammar Inference or Language studying) is the method of getting to know of a grammar from training information of the wonderful and poor language of strings. The PDA and CFG performance are the same. A CFG generates context-free language, and a PDA recognizes context-free language. A context-free grammar is made up of four parts: variables, terminals, a start variable, and rules. On the "left side," each rule must have a single variable, and on the "right side," it can have any combination of variables and terminals. A pushdown automaton is a finite state machine in which things can be pushed or popped off of an infinitely tall stack on each transition. This document explains how to develop a CFG into a PDA that accepts its language. A CFG might be used to define a programming language, and the PDA could be used to design the language's compiler, thanks to the equivalence. With Greibach Normal Form inputs, a context-free grammar to pushdown automaton converter is available. Pushdown automata is shown to identify the context-free language class. A finite automaton identifies every regular language.

Keywords— Finite Automata, Context Free Grammar, Pushdown Automata, Conversion.

# Introduction

Computer scientists work with a wide variety of languages and grammars that fall within the Chomsky hierarchy, which was first proposed by linguist Noam Chomsky in 1956. A context-free grammar (CFG) with a finite set of grammatical rules form a quadruple (V, T, P, S), where V is a variable (non-terminals). T stands for terminals. P is a set of instructions, P: V (V T) \*, i.e., the manufacturing policies' left-hand side. There is no appropriate or left context for P. The first image is S. Pushdown automata are a means to implement a CFG and a structure DFA for a standard grammar in the same way. A DFA can only retain a certain number of records, whereas a PDA can retrieve an infinite quantity of data.

This language categorization scheme is linked to the categorization of automata that may be used to comprehend such languages, as well as the grammars that are used to create them. Designing an automaton M that accepts as input sequences in is the challenge of detecting a language L over an alphabet. If sequences in are acceptable factors of the language L, M accepts them; otherwise, M rejects them. Each grade of language in the Chomsky hierarchy is created using a special type of grammar and can be identified using a specific type of automaton. As one moves up the language hierarchy, the type of automata essential to understanding the language grows more powerful, and the grammar required to create the language becomes more common. Fig. 1.

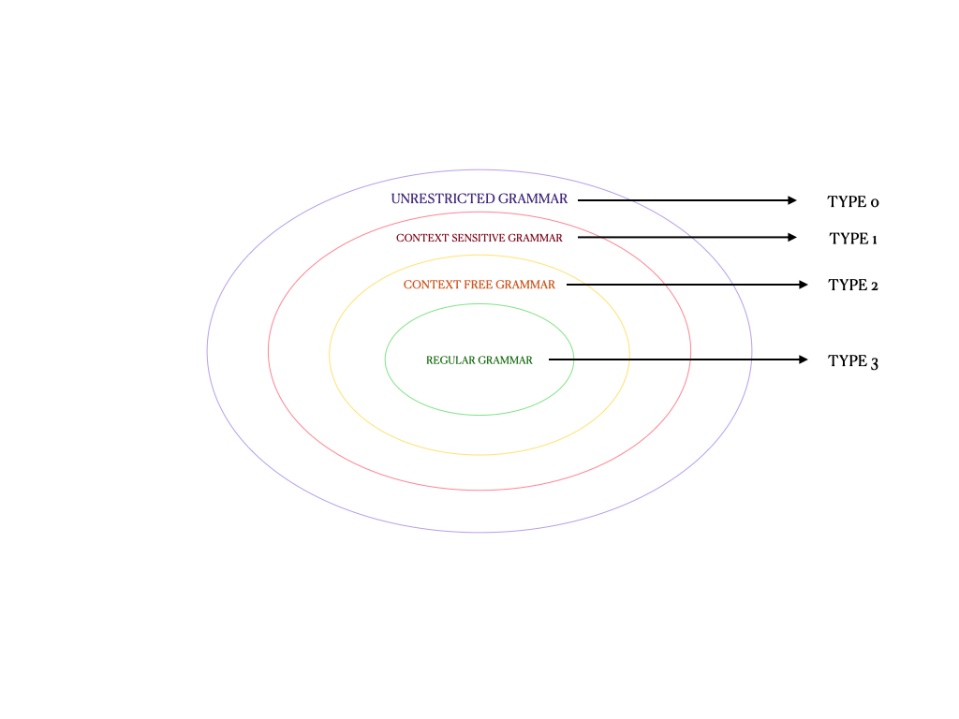


Fig. 1. Hierarchy of Chomsky

Illustration CFG languages are the kind of languages discovered using pushdown automata, which are finite automata with an indefinite stack. The Pushdown Automata (PDA) is depicted in Figure 2 as a block diagram.

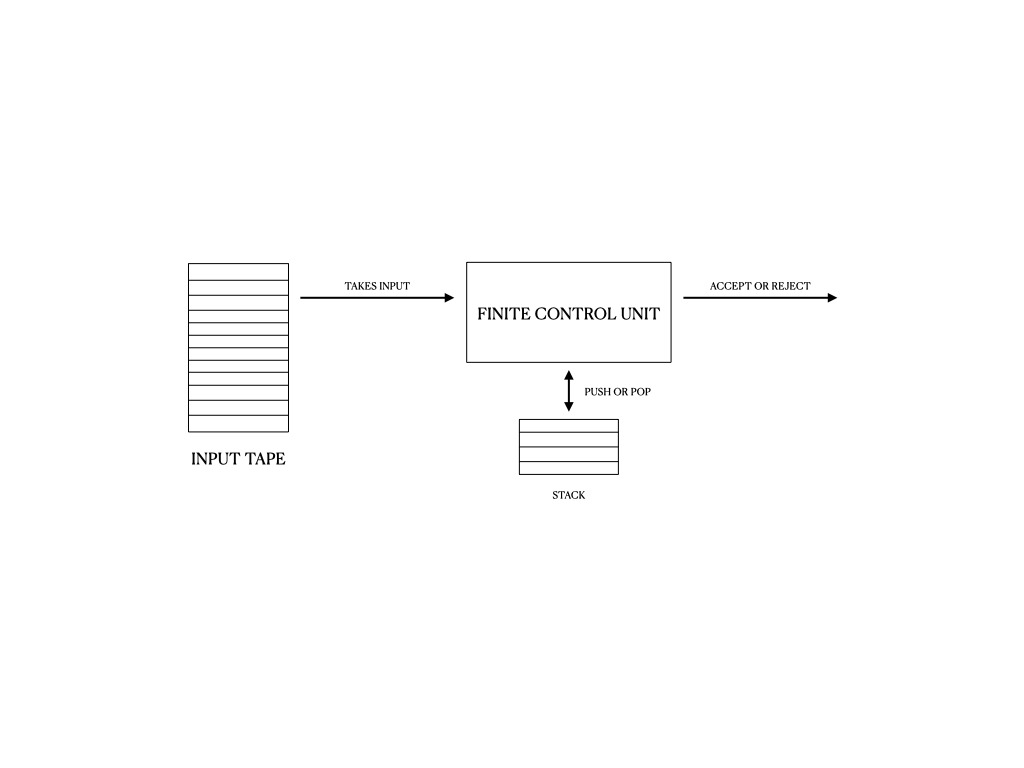


Fig. 2. Schematic diagram of the Pushdown Automata

Grammar induction is a method of learning grammar from educating facts about the language's fine and negative strings. The impact of several grammar representations found which an evolutionary method based on widely used context-free grammars (Backus-Naur form, BNF) outperformed their representations. CFG has become easier to investigate using evolutionary algorithms by providing grammar-specific heuristic operators and better populating the preliminary population.

# Literature Survey

Context Free Grammar is a notation that can be used to specify a language's grammar (CFG). Terminals, non-terminals, a start symbol, and production rules make up a Context Free Grammar. The terminal symbols are a group of tokens[5]. These are the fundamental symbols that make up strings. Non terminals are the sets of strings that are represented by syntactic variables. The collection of strings marked by the start sign is the grammar's stated language[5].Context free grammar, which is type 2 in Chomsky's tier has a lot of bandwidth available for applications in Bottom-up Processing, logical theory, recognition of language, machine learning, genetic engineering, and more [4].

Several papers have explained the benefits of using Context Free Grammars (CFGs) using proofs and Parse Trees to reduce uncertainty in Natural Language Sentences[5].These advantages, however, are contingent on the Derivation processes and CFG Rules.

The paper[6] presents a context-free grammar to limit the identification of a system's use cases in order to adhere to the requirements of engineering discipline. The suggested CFG will be used to represent and record the rules that control the identification of events (designated as syntactic categories), use cases, and other information system requirement components (designated as the set of valid terminals). It is envisaged that the number of errors in identifying use cases would be reduced.

Grammatical inference/grammar induction encompasses a wide range of subjects, including identifying unfamiliar Context Free grammar in the chosen language from positive examples[4].

In this study[4], a new method is presented for determining the Chomsky Normal form's equivalent. It is been proved that GFGness is ExpTimecomplete for properties other than PDA. VPA can be exponentially more concise than GFG-VPA, and GFG-VPA can be exponentially predictable VPA is more concise.

The topic of learning context-free grammar is investigated from a corpus in the paper [7]. A technique based on the concept of the corpus's minimum description length is been looked into. The cost as a function of grammar is the sum of the number of bits expected for the presentation of a grammar and the number of bits needed for the derivation of a corpus using that grammar.

Path inquiries are being used to specify paths inside a data network that conform to a specific arrangement [1]. Regular path patterns created by regular expressions are frequently supported by query languages like SPARQL. A context-free path query defines a path whose language could be determined by a context-free grammar[1].

This type of query can be used in a number of domains, including genomics, data science, and open source analysis. The approach for handling context-free path queries is presented in this paper. In contrast to existing systems that must process the entire graph, the algorithm mentioned searches for localised pathways, allowing it to process subgraphs. It also accepts any context-free grammar as input, avoiding the use of troublesome normal forms[1].It looks at the traditional correspondence theorem between context-free grammars and pushdown automata in this study[2]. Changing the sequential composition process operator to a sequence alignment operator with conditional acceptance improves the correlation in the scenario[2]. The missing factor in reconstructing the entire correspondence is revealed to be the addition of a concept of situation awareness [2].

In input-driven pushdown automata (IDPDA), the next activity on the pushdown store (push, pop, nothing) is solely controlled by the possible source. Nowadays, many similar devices are designed so that pushing from an unoccupied pushdown does not terminate the calculation, but rather maintains it [3].

The impact of DIDPDAs' novel behaviour on their power is investigated, and their capabilities are compared to those of input-driven pushdown automata, that are effectively IDPDAs with input pre-processed by duration finite state transducers [3], are two types of IDPDAs. It turns out that the powers are unrivalled. It is investigated that the determinization of DIDPDAs, as well as their sheer intricacy, closure features, and decidability [3].

By investigating, it can be said that these transformations from the perspective of descriptional complexity in this paper [9]. The ideal optimum values are presented for the majority of states of nondeterministic and deterministic finite automata that are comparable to Chomsky normal form unary context-free grammars [9]. The amount of elements within grammars in question determines the boundaries. There are additional upper limitations for the majority of states of finite automata that emulate unary pushdown automata [9].

As a result, it shows that one-way auxiliary pushdown automata's workspace for accepting non regular unary languages has a log log n lower bound. "Weak space" [9] is the idea of space the paper is aiming at.

SCFGs (synchronous context-free grammars) are increasingly widely used in statistical machine translation, with Hiero (Chiang 2007)[10] being the most well-known example. The task at hand was to use a SCFG and an n-gram language model to decode source text and generate a target translation [10].

In practice, decoding is challenging, yet the fundamental linguistics and relationships involved may be articulated simply and clearly. Because of the formal description, PDAs will be available for comparison to current decoders based on other forms of automata [10]..

# Common Information On the System

In this paper, with Greibach Normal Form inputs, a context-free grammar to pushdown automaton conversion is available. The paper used the following method to convert the CFG to PDA. On R.H.S. manufacturing, the initial symbol has to be a terminal sign.

The steps for obtaining PDA from CFG are as follows:

Step 1: Change the CFG outputs into GNF outputs.

Step 2: There will be only one state {q} on the PDA.

Step 3: The CFG's initial symbol will be the PDA's initial symbol.

Step 4: Add the appropriate rule to non-terminal symbols:

1. δ(q, ε, A) = (q, α)

Where the production rule is A → α.

Step 5: For each terminal symbols, add the following rule:

1. δ(q, a, a) = (q, ε) for every terminal symbol.

Fig 3: -CFG to PDA convertor [25]

# Advantages And Disadvantages of the Existing System

A string can be accepted in one of two ways by a PDA. A PDA recognizes a string, as per the first, when it achieves its final state after reading the entire string. Per the second, a PDA validates a string when it has read the entire string and has emptied its stack [13].

By defining the CFG to PDA transfer process, it may create an incremental parsing method for any CFG [13]. When the NPDA is approached with a nondeterministic decision, a retracing search can be prepared; if the NPDA is confronted with a nondeterministic decision, a piece of code is organised that iteratively hunts for one of the paths to approve [13].

The stacked alphabet of the PDA produced in the CFG to PDA conversion procedure is formed by the non-terminals and terminals of the supplied grammar [13].

It could be said that alike LSTM and Transformer networks visualise context-free languages with constrained iteration and similar representation power. However, the disadvantage of the LSTM model is that it fails to factorise its innate area to encrypt the state and numerous aspects of the stack without clear and specific guidance, which is the main pillar to its vulnerable results in real time projects and in practise [12].However, the lack of explicit breakdown normalisation has a slight effect on the Transformer [12]. Since the results are sensitive to the configurations of PDAs, the working practice of the models, and the standard operating procedures, language identification is not a realistic job to evaluate the factual ability of LSTM and Transformer [12].

# Proposed System

There is no given algorithm to determine if there exists a deterministic PDA for a grammar so we can actually find some result on this so that there will be some proper algorithm.

Also, PDA cannot concede environment-sensitive and unrestricted grammar, it can only allow context-free grammar so we need to find a way in which PDA will acknowledge context-sensitive and unrestricted grammar.

Every non-deterministic PDA cannot be converted to a deterministic PDA. Even if we have a nondeterministic PDA that is guaranteed to have deterministic originality to it so, there is no mechanical procedure to find it.

# Conclusion

An adequate fit of method graphs under language similarity is called a language. A pushdown automaton's set of languages matches the set of languages provided by either a context-free grammar.

Integration of automata theory and process theory is a project we're working on right now. As a result, we can use a computational model with interactivity to illustrate the basics of computer science. The use of formal languages theories to software engineering will aid in improving overall accuracy of the measurements indicated.

On the basis of several versions of minimization of conventional nondeterministic automata, minimal automata can be created.

One can create a separate expense function that takes symbol frequencies into consideration and reduces the amount of bits even.

##### References

1. Querying graph databases using context-free grammarsCiro M.Medeiros, Martin A.Musicante, Umberto S.Costa, 2022.*)*
2. PUSHDOWN AUTOMATA AND CONTEXT-FREE GRAMMARS IN BISIMULATION SEMANTICS , JOS C. M. BAETEN, CESARE CARISSIMO , AND BAS LUTTIK, 2022
3. Digging input-driven pushdown automata, Martin Kutrib and Andreas Malcher, 2021
4. Context Free Grammar Identification from Positive Samples, 1K.Senthil Kumar, 2D.Malathi, 2018
5. NLP : Context Free Grammars and Parse Trees for Disambiguating Telugu Language Sentences Jinka Sreedhar, Dr Sk Althaf Hussain Basha , D. Praveen Kumar, A.Jagan, Baijnath Kaushik, 2017
6. A Context Free Grammar for Requirements Modeling Michelle P. Banawan, 2012
7. Automatic Learning of Context-Free Grammar, Tai-Hung Chen, Chun-Han Tseng, Chia-Ping Chen, 2006
8. A Bit of Nondeterminism Makes Pushdown Automata Expressive and Succinct, Shibashis Guha, Isma¨el Jecker, Karoliina Lehtinen and Martin Zimmermann, 2021
9. Unary Context-Free Grammars and Pushdown Automata, Descriptional Complexity and Auxiliary Space Lower Bounds, Giovanni Pighizzini , Jeffrey Shallit, Ming-wei Wang, 2002
10. Pushdown Automata in Statistical Machine Translation, Cyril Allauzen, Bill Byrne, Adri`a de Gispert, Gonzalo Iglesias, Michael Riley, 2013.
11. PDA Simulator for CFG Induction Using Genetic Algorithm
12. Learning Bounded Context-Free-Grammar via LSTM and the Transformer:Difference and Explanations, Hui Shi, Sicun Gao, Yuandong Tian, Xinyun Chen, Jishen Zhao, 2021
13. Push-down Automata and Context-free Grammars.
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