CFG To PDA Converter using Python

Kuldeep Vayadande

*Artificial Intelligence and Data Science Vishwakarma Institute of Technology,* Pune,411037, Maharashtra. [kuldeep.vayadande1@vit.edu](mailto:kuldeep.vayadande1@vit.edu)

Anish Kamble

*Artificial Intelligence and Data Science Vishwakarma Institute of Technology,* Pune,411037, Maharashtra. [anish.kamble20@vit.edu](http://anish.kamble20@vit.edu/)

Aditya Kanawade

*Artificial Intelligence and Data Science Vishwakarma Institute of Technology,* Pune,411037, Maharashtra. [aditya.kanawade20@vit.edu](http://aditya.kanawade20@vit.edu/)

Sanika Kadam

*Artificial Intelligence and Data Science Vishwakarma Institute of Technology,* Pune,411037, Maharashtra. [sanika.kadam20@vit.edu](http://sanika.kadam20@vit.edu/)

Sanika Kadam

*Artificial Intelligence and Data Science Vishwakarma Institute of Technology,* P[une,411037, Maharashtra.](http://sanika.kadam20@vit.edu/)

Omkar Kale

*Artificial Intelligence and Data Science Vishwakarma Institute of Technology,* Pune,411037, Maharashtra. [omkar.kale20@vit.edu](http://omkar.kale20@vit.edu/)

Sejal Kharche

*Artificial Intelligence and Data Science Vishwakarma Institute of Technology,* Pune,411037, Maharashtra. [sejal.kharche201@vit.edu](mailto:sejal.kharche201@vit.edu)

***Abstract*— Grammar Induction (Language Studying) is a strategy for learning grammar by using training data from the language's good and bad strings. A CFG and a Pushdown Automata have the same ability. CFG generates Context-free language, and that language is discovered by a Pushdown Automata. Variables, terminals, a start variable, and rules are used to create a context-free grammar. On the "left aspect," and on the "right aspect," each rule should have only one variable. A PDA is an infinite-state machine that allows items to be pushed or popped off of an indefinitely tall stack at any time. This paper explains how to turn a Context Free Grammar into a Pushdown Automata that determines the language of the CFG. A context-free grammar to pushdown automaton converter is provided with Greibach Normal Form inputs. PDAs have been demonstrated to have CF language classes. Every regular language is recognized by a finite automaton.**

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

1. INTRODUCTION

Use of Context-Free Grammar is to generate all possible patterns of strings. This strings are given in a formal language. CFG is a formal grammar [6]. Formal language considers a language as a mathematical object. We can define Context-Free Grammar using four tuples, which are g = {(V(v), T(t), P(p), S(s))} [5].

Here, g stands for grammar, T stands for final set of ending symbol, V stands for final set of non-ending symbol, P stands for production rules and S stands for start symbol.

state that is at final, d: mapping function (It is used to move from current state to next state).

1. PROBLEM STATEMENT

Context-Free Grammar is used to construct all potential string patterns, as we saw earlier. These strings are expressed in a formal language, and PDA is a Non-Deterministic Finite Automata with external stack memory. Infinite amount of information can be remembered by PDA. But there is nothing that would help us to convert CFG to PDA.

Hence to solve this problem we need to find a way for PDA to recognize the context-sensitive and unconstrained alphabet, as it can only recognize the context-free alphabet and, we can't find way to transform every-deterministic PDA into a deterministic PDA. Indeed, if we’ve got a nondeterministic PDA that is guaranteed to have a deterministic original there is no mechanical procedure to find it.

As a result, we've supplied a method for converting a CFG into a PDA that determines the CFG's language and accepts Greibach Normal Form inputs.

1. LITERATURE REVIEW

Context-Free Grammar is a notation for describing the grammar of a language (CFG). Context-Free Grammar is made up of various terminologies [5]. Strings are made up of these fundamental symbols. The grammar's declared language is the collection of strings denoted by the start sign [5].

PDA stands for Pushdown Automata. Pushdown Automata is considered as NFA, additionally it is also increased with stack memory of external. Infinite amount of information can be remembered by PDA [10]. The name pushdown tell us about PDA that it can insert an element onto start of stack and can delete an element from the start of that stack. Pushdown automata is defined using seven components. This components are Q(q), ∑(s), Γ(T), qo, Z(z), F(f) and δ(d) [11].

Here, q: consists of the states, s: the input set, T: a stack

symbol, q0: the initial state, z: a start symbol, f: consists of

A number of research have discussed the advantages of utilizing CFGs to reduce uncertainty in Natural Language Sentences using proofs and Parse Trees [5]. However, these advantages are based on the Derivation procedures and Rules of the grammar.

The paper [6] proposes a context-free grammar for limiting the identification of a system's use cases while adhering to engineering discipline criteria. The proposed CFG will be used to express and record the rules that govern the identification of events (known as syntactic categories), use cases, and other information system requirements (designated as the set of valid terminals).

Grammatical inference/grammar induction covers a wide range of topics, including recognizing novel Context Free grammar [4].

A new approach for identifying the Chomsky Normal form's equivalent is described in this study [4].

In the work [7], the difficulty of learning the grammar is investigated using a corpus. A approach based on the minimal length of the description of corpus is being investigated.

Path queries are used to specify paths inside a data network that match a particular pattern [1]. Regular expression-based path patterns are frequently supported by query languages like SPARQL. A context-free path inquiry specifies a path whose language is determined by context- free grammar [1.]

Genetics, data science, and source code analysis all benefit from this type of inquiry. This paper describes a method for dealing with context-free path inquiries. Unlike other systems that must process the full graph, the technique discussed looks for localised paths, allowing it to process subgraphs. It also accepts any context-free grammar as input, which eliminates the need for cumbersome normal forms [1]. This study [2] investigates the classic correspondence theorem between CFG and PDA. The correlation in the situation improves when the sequential composition process operator is replaced with a sequence alignment operator with conditional acceptance [2]. The addition of a concept of situation awareness is found to be the missing piece in reconstructing the complete correspondence [2].

The succeeding activity on the pushdown store is exclusively controlled by the possible input in input-driven pushdown automata (IDPDA). Many such devices nowadays are built so that pushing from an unoccupied pushdown does not stop the calculation, but rather keeps it going [3].

DIDPDAs' new behavior is studied, and their capabilities are contrasted to pushdown automata which is driven by input, are effective IDPDAs with input pre- processed by duration transducers of finite-states [3]. The determinization of DIDPDAs, as well as their sheer intricacy, closure properties, and decidability, are examined [3].

This work [9] can be said to investigate these transformations from the perspective of discretional complexity. For the majority of states of NFA and DFA that are analogous to Chomsky's normal form unary context-free grammars [9], ideal optimal values are supplied. The bounds are determined by the number of items in the grammar in question [9].

As a result, the workspace for adopting non-regular unary languages in one-way auxiliary pushdown automata has a log log n lower constraint.

The most well-known example of SCFGs in statistical machine translation is Hiero (Chiang 2007) [10].

Although decoding is difficult in reality, the core languages and relationships involved may be expressed simply and plainly. PDAs will be compared to contemporary

decoders based on other types of automata [10] because of the formal description.

1. CFG AND PDA
2. FINITE AUTOMATA

Automata: Theory of automata is a theoretical department of science and mathematical. A theoretical mannequin of a laptop hardware or software program machine used in automata concept is regarded as summary computing device and this abstract computing device is referred to as Automata.

Finite Automata (FA): A finite automata is a five- element or tuple summary machine, also known as a finite state machine. Q(q), ∑(s), δ(d), q0 and F(f) (q: set of all the states, s: set of the image that is input, q0: initial state, f: closing state, d: Transition function) are the five tuples. Finite automata are used to understand patterns in general. Only conventional ordinary languages can be supplied to finite automata. It modifies the state of a string, which is made up of symbols. When the preferred symbol is discovered, the transition takes place.

Finite Automata (FA) has two states, they are “Accept” and “Reject” states. During transition the automata have two options, it can cross to subsequent country can remain in the same state. It will accept only when they enter completely successfully processed string, and the automata has reached its remaining state.

Finite automata are characterized into two types:

* Deterministic Finite Automata:

In DFA the computing device goes to a country just for an specific kind of input.

Transition design for DFA:

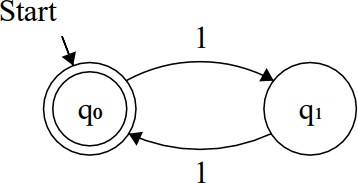


Fig. 1. DFA Diagram [29]

* Non-Deterministic Finite Automata:

In NFA the computer goes to one or many nations for a precise input.

Transition graph for NFA:

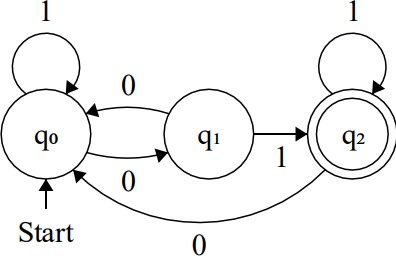


Fig. 2. NFA Diagram [29]

1. CONTEXT-FREE GRAMMAR

A CFG is a language description notation. Although it is more powerful than finite automata or REs, it still falls short of defining all possible languages. Parentheses in programming languages are a good example of nested structures. The basic concept is to utilize "variables" to represent groups of strings (i.e., languages). These variables are defined in terms of one another recursively. Only concatenation is involved in recursive rules ("productions"). Union is possible with alternative rules for a variable.

Grammar without context ‘g’ is defined using four tuples:

g = {(V(v), T(t), P(p), S(s))}.

The grammar is described by g.

A finite set of ending symbols are given by t.

A finite collection of non-ending symbols is described by v. Set of production guidelines is represented by p.

The letter s represents the beginning.

In CFG, the first term is utilized to create the string. The string can be replaced by non-ending terms on the production’s right side with ending terms till all the non- ending terms are changed.

Example: Construct a CFG for a language with any number of a's in set= Σ{a}.

L = {ε, a, aa, aaa, aaaa, }

CFG should be such that it should accept all strings. p = s  as …. (Recursive step)

s  ε

p can also be written as s  as| ε

p stands for the set of production of this grammar.



Fig. 3. Tree parsed for ε

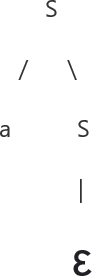


Fig. 4. Tree parsed for ‘a’

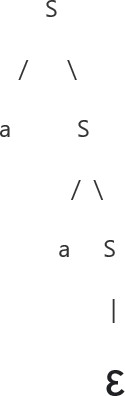


Fig. 5. Tree parsed for ‘aa’

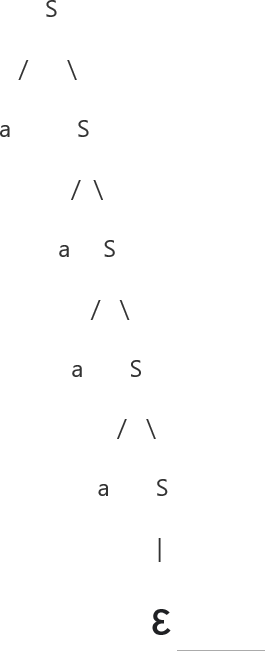


Fig. 6. Tree parsed for ‘aaaa’ CFG constructed is: g = ({s}, {a}, p, s).

Example: Construct CFG for the language having any number of c and d.

R.E. = (c + d) \*

L = {ε, c, d, cc, dd, ccc, ddd, cdcd, }

p = s  cs | ds | ε

i.e., s tends to cs | ds | ε

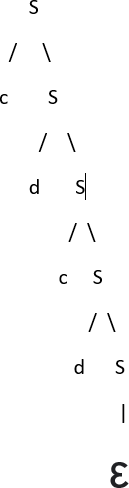


Fig. 7. Parse Tree for ‘abab’

CFG constructed is as follows: g = ({s}, {a}, {b}, p, s).

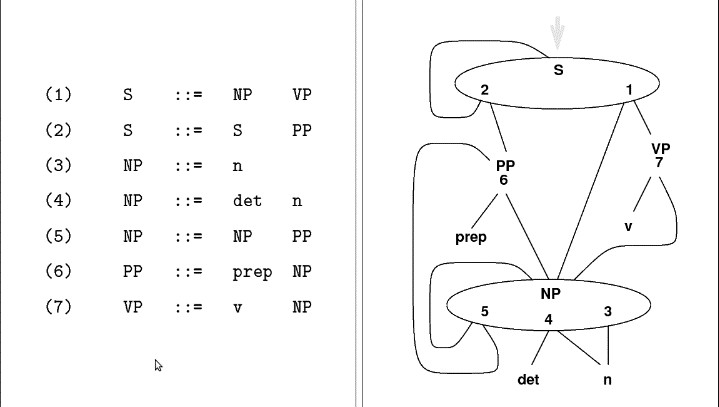


Fig. 8. Graph of the adjacent grammar [26].

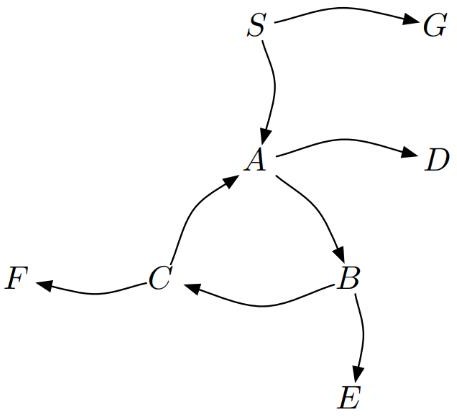


Fig. 9. Dependency graph of A CFG [30]. From the above graph (Fig. 11), we conclude that

– Nrec = {A, B, C};

– Nleech = {S};

– Nres = {D, E, F, G}.

1. PUSHDOWN AUTOMATA

Context-free languages, such as the set of regular languages, are acknowledged by pushdown automata. A context-free language is one that specifies strings with matching parenthesis. Assume a programmer has created some code, and any parentheses must match for the code to be valid.

One method is to feed the code (as strings) into a pushdown automaton with transition functions that implement the context-free grammar for the balanced parentheses language. The pushdown automaton will "accept" the code if it is legitimate and all parenthesis matches. The pushdown automata will be able to inform the programmer that the code is invalid if there are uneven parenthesis. One of the more speculative concepts underlying computer parsers and compilers is this.

When thinking about parser design and any other area where context-free grammars are employed, such as computer language design, pushdown automata can be useful. Because pushdown automata have the same computational capability as context-free languages, there are two ways to demonstrate that a language is context-free: give a CFG or a PDA for the language.

A Pushdown Automata (PDA) is described like follows:

* + Q(q) stands for finite set of all the states.
  + The automata is started from qo.
  + All the symbols with the inputs are in the set of ∑

(s).

* + The symbols that can be inserted and deleted the stack are in the set of Γ(T).
  + F(f) consists of all the states that are at the final .
  + Z(z) stands for the pushdown symbol, which is already present in the stack.
  + δ (d) converts q x x, into q x \*. PDA accepts symbol of input and symbol of stack (top of the stack) in one state and then transition to another state and alter the stack symbol.

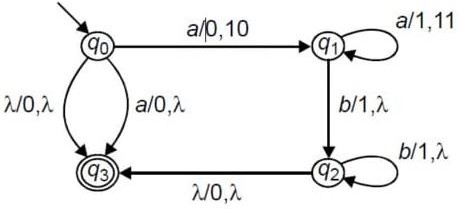


Fig. 10. Graphical Notation of a PDA [27].

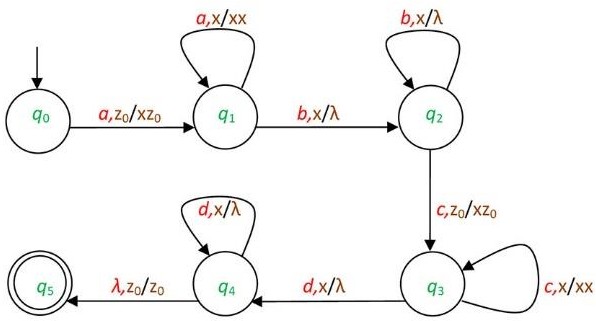


Fig. 11. Graphical Notation of a PDA [28].

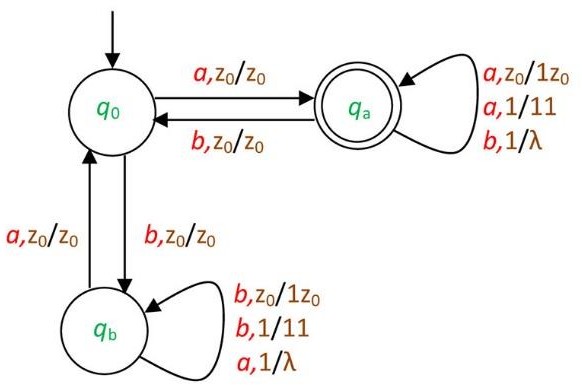


Fig. 12. Graphical Notation of a PDA [28].

The PDA is deterministic as the reason is, there's only one transition across states on a possible input and stack symbol. In non-deterministic PDA, numerous moves from a state on a data item and symbol of stack are possible.

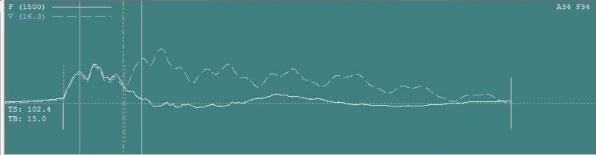


Fig. 13. Dynamic analysis 1 [31]

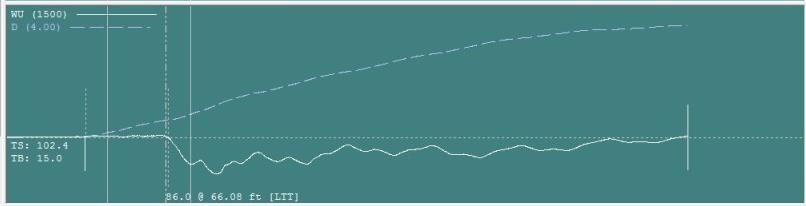


Fig. 14. Dynamic analysis 2 [31]

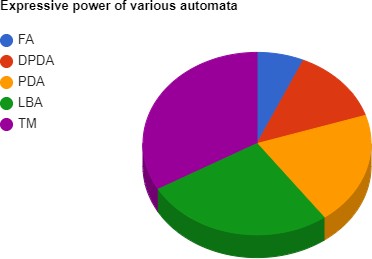


Fig. 15. Expressive Power of various automata

Chart

Description automatically generated

Fig. 16. Big DNA dataset analysis under PDA

The expressive capability of non-deterministic PDA is greater than that of expressive deterministic PDA. NPDA accepts some languages, whereas deterministic PDA does not.

Push down automata can be built using either acceptance by empty stack or acceptance by end state, and either can be converted to the other.

1. DATA ANALYSIS

CFG is a type of grammar. It may be the official language system used to capture all the string patterns within a given formal language. The grammatical grammar G is usually given with four leaflets as follows:

g = {(V(v), T(t), P(p), S(s))}.

There,

The g is the grammar, which includes the some rules of consolidation.

t specifies the final set of the term mark. It is described in lower case.

v represents the finite set.

p represents a set of production rules, generally used to replace non-terminal symbols.

s represents the start variable which is used to point out the whole program.

CFG contains some productions rules, which are used to convert the grammar to the PDA. e.g.

s

a, b

s--> asX|bsY|b

X--> ba

Y--> ab

In this way the grammar is represented and stored in the text file. Then we give the path of the file to the python file, and it converts this grammar into the Push Down Automata.

CFG to PDA conversion:

CFG is defined using four tuples so the input will be any of the four tuples {g = (v,t,p,s)}

PDA is define using Seven tuples so the output will be any of these seven tuples {P = (q, s, z, T, qo,, f, d )}

Following are some steps to convert CFG to PDA:

Step 1 − Step 1 is to transform the values of the (CFG) into (GNF).

Step 2 − The PDA would be having a {q} i.e., single state.

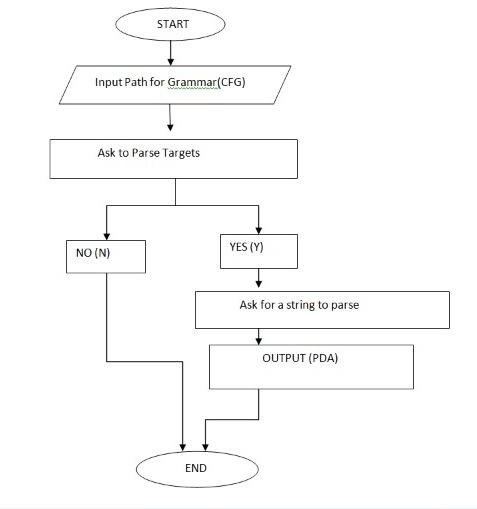
Step 3 − The first element that is input of CFG is the first element.

Step 4 −In PDA the stack symbols will be non-ending terms of the CFG, and the input symbols will be ending terms of the CFG.

Step 5 − For every value which is in A → aX(a is an ending term and A, X are combination of ending and nonending terms, make a transition d (q, a, A

1. OUR APPROACH

We need to discover a solution for PDA to recognize the context-sensitive and unconstrained alphabet, as it can only recognize the context-free alphabet. There is no way to transform an every-deterministic PDA into a deterministic PDA. Indeed, if we’ve got a nondeterministic PDA that is guaranteed to have a deterministic original there is no mechanical procedure to find it.

Fig. 16. Flowchart of the project CONCLUSION

A language is defined as an appropriate fit of method graphs under linguistic similarity. The collection of languages supplied by either CFG and a pushdown automaton are identical. We're now working on a project that combines process theory and automata theory. As a result, we may be able to use an interactive computational model to illustrate computer science basics. The application of formal language theories to software engineering will improve the overall accuracy of the measures. Minimal automata can be formed by combining different methods of minimization of standard nondeterministic automata.

OUR CONTRIBUTION

We have already discussed finite automata. Only regular languages can be accepted using finite automata. PDA is a finite automaton containing more memory for stack that allows it to detect CFL.

To convert context-free grammar to pushdown automata is a manual process before, but we can convert the same CFG to PDA using the python program in this proposed system. The given program will ask the user to parse targets Yes or No. Then the program asks the user to enter a string to parse. Ten given strings will be compared to a given language is the string in each language or not.

FUTURE WORK

A human reader is unconcerned about difficulties like set finiteness, which must be addressed directly in a theorem prover. The type of specification (relations vs. functions) has a significant impact on both the size of the proof and the ease with which it can be automated. A pushdown stack, which

significantly improves the language recognition capabilities when compared to other parsing algorithms employs them. The PDA can store an endless quantity of data. The sole disadvantage is that the stack data structure consumes more memory. The NL-PDA can be built and implemented for a variety of more natural languages in the future. The ELR- PDA has some limitations, such as the inability to interpret idioms and poetry language, but we will overcome this issue in the future.

Currently, our project is only working on Greibach Normal Form (GNF) in the future work we are trying to convert the CFG inputs which are in both (GNF) and (CNF). Also currently, in the given proposed system, the CFG is given as an input in a python program, but in the future, we will add handwritten recognition for CFG so that it will directly recognize through CFG written on paper.

ACKNOWLEGEMENT

Our guide, Dr. Kuldeep Vayadande, was instrumental in making this project model and paper viable. Because of the active engagement of all of our co-workers, we were able to complete this project model. Finally, we want to express our gratitude to the institution and university for their unwavering support and for providing us with the opportunity to further our education.

REFERENCES

1. Querying graph databases using context-free grammars, Ciro 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

1. Digging input-driven pushdown automata, Martin Kutrib and Andreas Malcher, 2021
2. Context Free Grammar Identification from Positive Samples, 1K.Senthil Kumar, 2D.Malathi, 2018
3. 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
4. A Context Free Grammar for Requirements Modeling Michelle P.

Banawan, 2012

1. Automatic Learning of Context-Free Grammar, Tai-Hung Chen,

Chun-Han Tseng, Chia-Ping Chen, 2006

1. A Bit of Nondeterminism Makes Pushdown Automata Expressive and Succinct, Shibashis Guha, Isma¨el Jecker, Karoliina Lehtinen and Martin Zimmermann, 2021
2. Unary Context-Free Grammars and Pushdown Automata, Descriptional Complexity and Auxiliary Space Lower Bounds,

Giovanni Pighizzini , Jeffrey Shallit, Ming-wei Wang, 2002

1. Pushdown Automata in Statistical Machine Translation, Cyril Allauzen, Bill Byrne, Adri`a de Gispert, Gonzalo Iglesias, Michael Riley, 2013.
2. PDA Simulator for CFG Induction Using Genetic Algorithm
3. Learning Bounded Context-Free-Grammar via LSTM and the Transformer:Difference and Explanations, Hui Shi, Sicun Gao,

Yuandong Tian, Xinyun Chen, Jishen Zhao, 2021

1. Push-down Automata and Context-free Grammars.
2. 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.
3. 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.

1. 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.
2. VAYADANDE, KULDEEP. "Simulating Derivations of Context- Free Grammar." (2022).
3. 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.
4. 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.](http://www.IJARIIT.com/)
5. 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](http://www.irjet.net/)

1. 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.
2. 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/](http://www.ijariit.com/)

1. 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.
2. 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
3. CFG to PDA Converter [https://www.chegg.com/homework-help/questions-and-](https://www.chegg.com/homework-help/questions-and-answers/draw-pda-grammar-convert-cfg-addressed-pda-p-q-p-q-0-1-g-x-z0-q-f-given-q16290065)

[answers/draw-pda-grammar-convert-cfg-addressed-pda-p-q-p-q-0-](https://www.chegg.com/homework-help/questions-and-answers/draw-pda-grammar-convert-cfg-addressed-pda-p-q-p-q-0-1-g-x-z0-q-f-given-q16290065)

[1-g-x-z0-q-f-given-q16290065](https://www.chegg.com/homework-help/questions-and-answers/draw-pda-grammar-convert-cfg-addressed-pda-p-q-p-q-0-1-g-x-z0-q-f-given-q16290065)

1. <https://conf.ling.cornell.edu/kgrove/mcfgcky/mcfgcky002.html>
2. [https://er.yuvayana.org/pushdown-automata-representation-with-](https://er.yuvayana.org/pushdown-automata-representation-with-solved-examples/#%3A~%3Atext%3DGraphical%20Notation%20of%20pushdown%20automata%20(PDA)%3A%2C-Pushdown%20automata%20are%26text%3DThe%20graphical%20representation%20of%20the%2Caccepting%20as%20for%20finite%20automata) [solved-](https://er.yuvayana.org/pushdown-automata-representation-with-solved-examples/#%3A~%3Atext%3DGraphical%20Notation%20of%20pushdown%20automata%20(PDA)%3A%2C-Pushdown%20automata%20are%26text%3DThe%20graphical%20representation%20of%20the%2Caccepting%20as%20for%20finite%20automata) [examples/#:~:text=Graphical%20Notation%20of%20pushdown%2](https://er.yuvayana.org/pushdown-automata-representation-with-solved-examples/#%3A~%3Atext%3DGraphical%20Notation%20of%20pushdown%20automata%20(PDA)%3A%2C-Pushdown%20automata%20are%26text%3DThe%20graphical%20representation%20of%20the%2Caccepting%20as%20for%20finite%20automata) [0automata%20(PDA)%3A,-](https://er.yuvayana.org/pushdown-automata-representation-with-solved-examples/#%3A~%3Atext%3DGraphical%20Notation%20of%20pushdown%20automata%20(PDA)%3A%2C-Pushdown%20automata%20are%26text%3DThe%20graphical%20representation%20of%20the%2Caccepting%20as%20for%20finite%20automata) [Pushdown%20automata%20are&text=The%20graphical%20repres](https://er.yuvayana.org/pushdown-automata-representation-with-solved-examples/#%3A~%3Atext%3DGraphical%20Notation%20of%20pushdown%20automata%20(PDA)%3A%2C-Pushdown%20automata%20are%26text%3DThe%20graphical%20representation%20of%20the%2Caccepting%20as%20for%20finite%20automata) [entation%20of%20the,accepting%20as%20for%20finite%20autom](https://er.yuvayana.org/pushdown-automata-representation-with-solved-examples/#%3A~%3Atext%3DGraphical%20Notation%20of%20pushdown%20automata%20(PDA)%3A%2C-Pushdown%20automata%20are%26text%3DThe%20graphical%20representation%20of%20the%2Caccepting%20as%20for%20finite%20automata) [ata](https://er.yuvayana.org/pushdown-automata-representation-with-solved-examples/#%3A~%3Atext%3DGraphical%20Notation%20of%20pushdown%20automata%20(PDA)%3A%2C-Pushdown%20automata%20are%26text%3DThe%20graphical%20representation%20of%20the%2Caccepting%20as%20for%20finite%20automata)
3. <https://gyires.inf.unideb.hu/GyBITT/14/ch04s06.html>
4. <https://www.cs.unc.edu/~otternes/comp455/2-finite-automata.pdf>
5. Naive Infinite Enumeration of Context-free Languages in Incremental Polynomial Time Christophe Costa Florˆencio, Jonny Daenen, Jan Ramon, Jan Van den Bussche, Dries Van Dyck.
6. <http://www.acspile.com/dynamic-pile-analysis/pda/pda-graph/>
7. [https://www.researchgate.net/figure/Bar-graph-illustrating-number-](https://www.researchgate.net/figure/Bar-graph-illustrating-number-of-PDA-surgeries-undertaken-by-cardiothoracic-centre_fig2_352259092) [of-PDA-surgeries-undertaken-by-cardiothoracic-](https://www.researchgate.net/figure/Bar-graph-illustrating-number-of-PDA-surgeries-undertaken-by-cardiothoracic-centre_fig2_352259092) [centre\_fig2\_352259092](https://www.researchgate.net/figure/Bar-graph-illustrating-number-of-PDA-surgeries-undertaken-by-cardiothoracic-centre_fig2_352259092)