Imprecision in n-Parallel Right Linear Grammar and its application

Thesis Report

submitted in partial fulfillment of the requirements

for the award of degree of

Master of Engineering

in

Computer Science and Engineering

Submitted By

Aatma Vibhor Mishra

(801532001)

Under the supervision of:

Dr. Ajay Kumar

Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

July 2017

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "Imprecision in n-parallel right linear grammar and its application", in partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Ajay Kumar and refers other researcher's work which are duly listed in the reference section. The matter presented in the thesis has not been submitted for the award of any other degree of this or any other University.

Amishra

(Aatma Vibhor Mishra)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

A Kumar

(Dr. Ajay Kumar)

Assitant Professor, CSE Department

Acknowledgement

I would like to express my thanks to my guide Dr. Ajay Kumar for being an excellent mentor for me during my whole course of the thesis. His encouragement and valuable advice during the entire period have made it possible for me to complete my work.

He gave me complete freedom, right from choosing the topic for thesis until its completion. Timely checks and guidance about how to improve the work done played an integral role. Also, his support and patience in times of difficult situations helped me recover when my steps went astray.

Next most importantly all this would not have been possible without the patience and support of my family. Their constant sense of care encouraged me to complete my work.

I would like to acknowledge Dr. Maninder Singh for setting high standards for his students and encouraging them time and again to achieve them as well.

Aatma Vibhor Mishra

Abstract

Various types of grammar such as regular, context-free, context-sensitive and recursive enumerable grammar exists. All these grammars have their acceptance power which is limited to a certain domain only. Context free grammars are not able to display all features of natural languages and context sensitive grammars have many limitations like that of membership problem, emptiness problem. To overcome these dependencies, the concept of regulated grammar has been introduced. What regulated grammar basically does is to put some types of regulations on the context free grammar rules. In this way we take context free rules, apply some regulation on some rules so that if one rule is used then it also necessary that the bounded rules must be used along with it. The number of bounded rules depends on the problem statement. There can be as many number of production rules which can be regulated. In this way we can solve many problems which previously used context sensitive grammar, for them we can build regulated grammar.

We have used the concept of fuzziness on n-parallel right linear grammar. Previously when we have used the grammar to generate a certain string, either that string is completely generated or it is not. With the addition of fuzziness in the grammar we can also accept a string which was previously not accepted can now be accepted with a degree of membership. This degree of membership for erroneous string lies between 0 and 1. The degree of membership for a particular production is determined on the basis of data sets. There is different degree of membership for different kind of rules, whether it is for deletion of an alphabet or addition of a new alphabet or substitution of a new alphabet. We have applied the concept of regulated automata on the Chinese tones and constructed a grammar for the same.

Table of Contents

Certificate	I
Acknowledgement	ii
Abstract	iii
Table of Contents	iv
List of Figures	vi
List of Tables	vii
List of Keywords	viii
Chapter 1: Introduction	1-8
1.1 Regulated Automata	1
1.2 Preliminaries	2
1.2.1 Finite Automata	3
1.2.2 Chomsky Heirarchy	4
1.3 Regulation on Grammar	5
1.3.1 Context Based Regulation	6
1.3.2 Rule Based Regulation	7
1.4 Fuzzy Sets	7
Chapter 2: Literature Survey	9-19
2.1 Related Work	9
2.2 Types of Regulated Automata	9
2.2.1 Self-regulating finite automata	9
Chapter 3: Problem Statement And Proposed Solution	20
3.1 Gap Analysis	20
3.2 Objectives	20

3.3 Proposed Solution	20
3.3.1 Methodology	20
Chapter 4: Proposed Algorithm	21-27
4.1 Applying Fuzziness on the Grammar	21
Chapter 5: Results	28-34
5.1 Mathematical Model for representation of Chinese Tonal Inversion Symmetries	28
5.2 Self-regulated automata for Chinese Tonal Symmetries	29
5.3 n-Right Linear Simple Matrix Grammar for Chinese Tonal Symmetries	32
Chapter 6: Conclusion And Future Scope	35
6.1 Conclusion	35
6.2 Future Scope	35
References	36-39
Research Publications	40
Video Link	41
Plagiarism Report	

List of Figures

Figure No.	Description	Page No.
1.1	Chomsky Hierarchy	5
1.2	Different types of regulated grammars	6
2.1	n-first self-regulating finite automata	11
2.2	1-turn all self-regulating finite automata	12
5.1	Example of grammatical string for inversion.	29
5.2	Self-Regulated Finite Automata for Chinese Tonal symmetry for Inversion Rules	30
5.3	Reading 1st symbol Ping Self-regulated finite automata	30
5.4	Reading 2nd symbol Ping using self-regulate finite automata	31
5.5	Reading Ping after reading 4th symbol using self-regulated finite automata	31
5.6	Reading 6th symbol Ze using Self-Regulated Finite Automata	32
5.7	Reading last symbol Ping using self-regulated finite automata	32
5.8	Parse tree after applying Rule 1	33
5.9	Derivation tree after applying rule 2	33
5.10	Derivation tree for string *w* is represented in.	34

List of Tables

Table No.	Description	Page No.
1.1	Showing various types of grammar using Chomsky hierarchy	4
2.1	Literature Summary	14
2.2	Types of Regulated Grammar	16
2.3	Types of Regulated Automata	18

List of Keywords

CFG	Context Free Grammar
CSG	Context Sensitive Grammar
SFA	Self-Regulated Finite Automata
FSFA	First Self-Regulated Finite Automata
ASFA	All Self-Regulated Finite Automata
PRLG	Parallel Right Linear Grammar

Chapter 1
Introduction

1.1 Regulated Automata

In automata, we study the abstract model of machines as well as we try to solve many computational problems that can be solved using these machines. Automata combine the concept of mathematics with the concept of discrete mathematics. It represents a formal language which may be a finite or infinite set, which depends on what that set it is representing. In a more informal sense, the automata take some input, which is called as input alphabet and then moves from state to state. This transition depends on the transition function which is defined for that automata. The automata can either accept or reject the input. Automata play a very wide role in various areas like artificial intelligence, parsing, compiler design, natural language processing, formal verification, the theory of computation and much more.

Various modified version of the automata exists. These modifications include many modifications which can be done on input alphabet or states or transition function or acceptance condition. The input can be infinite or finite, or it can be tree word input or infinite tree input. The states can be finite, infinite or stack memory which is used in push down automata. Transition function can be deterministic, non-deterministic or there can be an alteration in the input alphabet. Acceptance can also be altered. It can accept finite words or infinite word or there can be a probabilistic acceptance.

All of the grammars mentioned above were able to perform various tasks, but they also have their limitations. Regular languages were able to recognize simple strings but didn't have any context based intelligence. That's when the concept of grammars came in which we also used the concept of contexts in their rules. There were these context free grammars which were able to replace the symbols without context. Context free grammars were not able to represent all aspect of natural languages. They were able to perform various tasks in many fields, but there was a necessity where the symbols have to be replaced based on its context. The good thing about context sensitive grammar is that they were able to represent many aspects of natural languages. Context sensitive grammars have many dependencies. These dependencies can be of interleaved type or nested type. The context sensitive grammars had many problems like the one of membership problem and so many other also. To overcome

these problems, we have introduced the concept of regulated grammar. The regulated grammars overcome this difficulty of context free grammar and context sensitive grammar by putting restrictions on the context free grammars rules. What regulated grammars did was used the context free rules of the grammars and applied some regulations on these grammars rules. The regulations are defined in such a way that if one rule is applied then other rules which are bundled with that rule will also be applied. This puts a restriction on the use of rules. The no of rules to be applied in regulation depends on the application. In this way, we get both the benefit of context free grammars and context sensitive grammars without worrying about their complexities and limitations.

1.2 Preliminaries

This section covers the basic concepts of automata, which includes many concepts like what is a finite language, alphabet, string, finite automata. This also includes various kinds of grammar which include regular grammar, context free grammar, context sensitive grammar. It also includes Chomsky hierarchy which shows the relation between various grammars and shows their acceptance power and relation among themselves in a very neat way. [12,15]

A finite language is a collection of strings where the strings are formed based on some conditions. This language varies from condition to condition. The condition depends upon for what the language is designed for. The following language produces the same number of a's and b's. It is represented below in a set form. $L = \{a^n b^n | n \geq 1\}$. Few strings which can be generated from this language is $\{ab, aabb, aaabbb \ldots\}$. $L = \{a^n b^m | n.m \geq 1\}$ language produces a certain string of a's and b's and there is no restriction on a's and b's (there's no condition in between them). Few strings that can be generated form this language are $\{ab, aab, abb \ldots\}$.

The alphabet is any finite non empty set of symbols. These symbols are used for input in the automata. By using a combination of these symbols, a string is generated, which can be accepted or rejected by the automata. Few alphabet sets ae shown below. $\Sigma = \{1,2,3,4\}$ and $\Sigma = \{a, b, c \ldots z\}$. Alphabets are used to generate strings which is any finite sequence of symbols for the given alphabet. There can also be a string which does not have any alphabet, it is called as empty string which is denoted by a Greek letter ϵ.If we consider a input alphabet string $\Sigma = \{0,1\}$ then few of the strings that can be generated using this input alphabet is $\{0,01,101,011, \epsilon\}$.

1.2.1 Finite Automata

It is a finite state machine which either accepts or rejects strings or runs that automata for a particular unique strings, which follows a special property of that automata. It is a mathematical model which contains finite no of states and transitions defines between them. It can be only in one state at a given moment of time. Input is used to change the state of the machine. The computational power of the finite automata is less than comparatively to pushdown or Turing machines. It is best used when we have to use when the states are predetermined. There are two kinds of finite automata one is DFA, and another one is NFA. DFA is the deterministic version of the finite automata while the NFA is the non-deterministic version of the finite automata [8,18].

Def. 1 [8]: A finite automaton contains five tuples $M = (Q, \Sigma, \delta, q_0, F)$ where, Q is finite no of states, Σ is input alphabet, δ is transition function which is defined as In FA: { $\delta: S \times \Sigma \to S \; for \; DFA \mid \delta: S \times \Sigma \to P(S) \; for \; NFA$ }, q_0 is the initial state and F is the set of final states.

Def. 2 [8]: Regular grammar has four tuples $G = (N, \Sigma, P, S)$ Where, N is set of non-terminals, Σ is the set of input alphabet, P is the set of production rules, S is the starting symbol. A regular grammar always describes a regular language. It has a set of production rules which generate the strings which are meant to be produced by these grammars. They can be either left linear or right linear.

Def. 3 [2]: Context free grammar is a set of rules to generate strings which are without context. Production rules are simple replacements. The rules are applied regardless of the context. We can have many context free grammars which generate the same context free language. Context free language can be used to show the structure of the natural languages. It consists of a finite set of grammar rules and has four tuples $M = (N, T, P, S)$. Where, N is set of non-terminals where $N \cap T = \phi$, T is set of terminals, P is the set of production rules, the left hand side of the production rule does not have any context and S is the starting symbol.

Example 1.1: Consider a context free language L given by ({A, {a, b, c}, P, A) where P: $A \to Aa, A \to abc$.

Def. 4 [8]: Context sensitive grammars generate context sensitive languages which are of the form $\alpha A \beta \to \alpha \gamma \beta$. Where A is a Non terminal $A \, \epsilon N$ and $\alpha \beta \gamma \epsilon (T \cup N)^*$ (ie: Strings of terminals and non-terminals). Here α, β can be empty but γ must be non-empty. The rule $S \to \varepsilon$ is allowed if S does not appear on the right hand side of any

rule. The languages generated by this grammar is recognized by linear bounded automaton. It allows replacement of A by γ only in the context of $\alpha - \beta$. Context sensitive grammars are used in so many areas where context is taken into consideration like that or natural languages. The subject verb agreement in the languages can be represented by context sensitive grammars [25].

Example 1.2: Consider context-sensitive grammar for language L $G\{\{S, A, B, C, a, b, c\}, \{a, b, c\}, P, S\}$ where P is the set of rules

$$S \rightarrow aSBC$$

$$S \rightarrow aBC$$

$$CB \rightarrow BC$$

$$aB \rightarrow ab$$

$$bB \rightarrow bb$$

$$bC \rightarrow CC$$

The language generated by this grammar is $\{a^n b^n C^n | n \geq 1\}$.

1.2.2 Chomsky Hierarchy

Nom Chomsky has shown the hierarchy of different formal languages. This hierarchy can show the expressive power of different languages classes. Which also shows how one class of language is contained in another class of language. The table below shows the different class of language families and with respect to the language families what kind of automaton they use.

Table 1.1: Showing various types of grammar using Chomsky hierarchy [26]

Grammar Type	**Grammar Accepted**	**Language Accepted**	**Automaton**
Type 0	Unrestricted Grammar	Recursively enumerable Language	Turing Machine
Type 1	Contest Sensitive Grammar	Context sensitive language	Linear Bounded Automata
Type 2	Context Free Grammar	Context free language	Pushdown Automata
Type 3	Regular Grammar	Regular Language	Finite state automaton

The kind of grammar they accept with respect to the type of the grammar it uses. According to this hierarchy, Type-0 grammar has the biggest acceptance power, and it includes all other language families. According to it, the Turing machine is the most powerful mathematical model which can be used for computation [26].

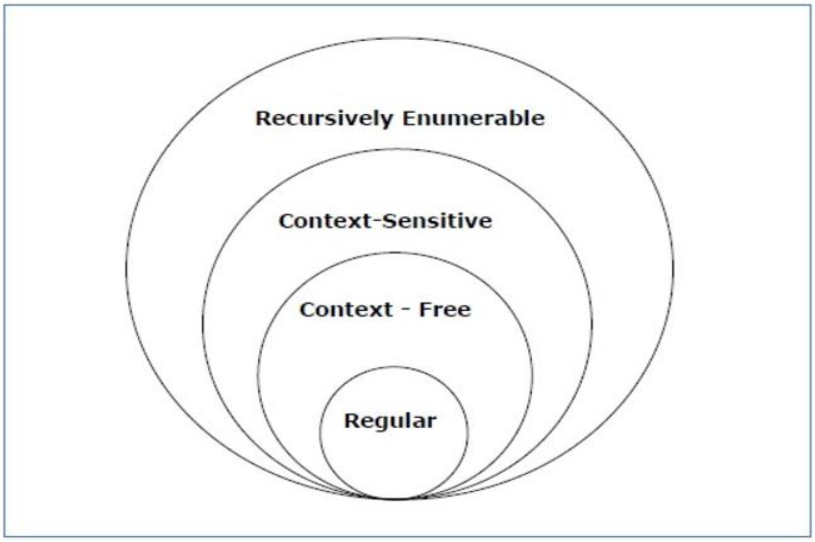


Fig 1.1: Chomsky Hierarchy [8]

The above figure shows the various scope of different types of grammar and their subset and superset. The grammars define above were able to do many things depending on their power, but few grammars above have certain restrictions which were removed by new grammars. One such limitation is put by context sensitive grammars where the symbol is replaced with some context, not just blindly. But it turns out these automatons were hard to make, and other functions on it were not so feasible, so to make an easy alternative we have introduced regulated grammar. Regulated grammar can do similar things a context sensitive grammar does, and they do it with more ease.

1.3 Regulation on Grammar

Regulated grammar is a type of grammar where we place a certain kind of regulation on the grammar rules. This regulation can be applied in two ways. It can be either context based regulation or it can be rule based regulation.

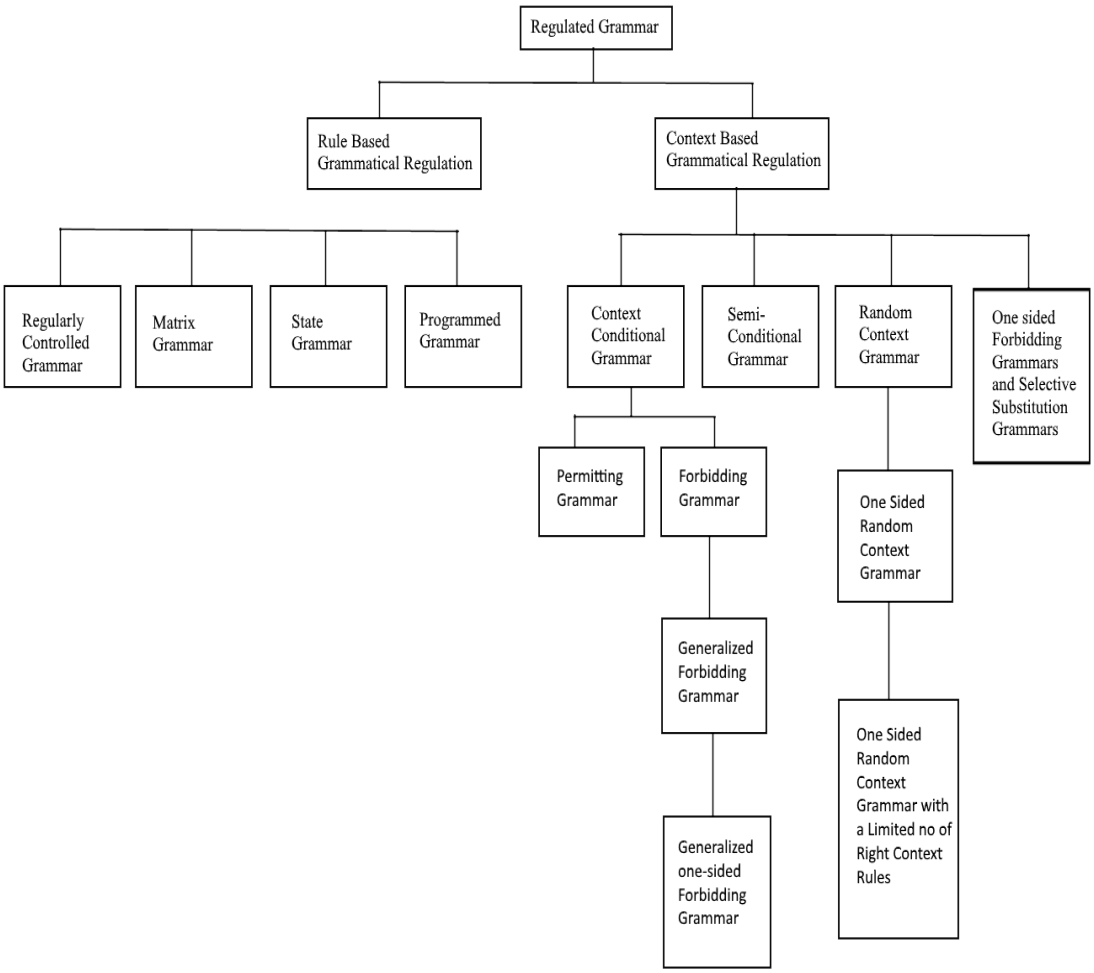


Fig 1.2: Different types of regulated grammars [2]

1.3.1 Context based regulation

What we have seen in context sensitive grammar and phase structure grammar is that they naturally are of types where in their left side they have string, not a single variable. The symbol is replaced by another symbol using that particular context of the symbol in its neighbourhood, but in scattered context regulation they drop the condition of neighbourhood symbol.

Def 5 [1,30]: Context conditional grammar uses context free regulation which can be finitely extended by either permitting or forbidding strings. It has four tuples $V = (V, T, P, S)$. Where V,T,S are total alphabet and terminal alphabet ($T \subset V$) and $S \in V - T$ and productions are of the form $P: (A \to x, Per, For)$, Where $A \in (V - T)$, $x \in V^*$ and $Per, For \subseteq V^+$.

Def 6 [1,30]: Random context grammars are three special cases of where the condition is applied on non-terminal symbols as a degree does not exceed (1,1).

Def 7 [1,30]: Generalized Forbidding grammar forbids the context conditions formed by finite languages. G = (V, T, P, S) be a context conditional grammar G. If For every $(A \to x, Per, For) \in P$ satisfies $Por = \emptyset$.

Def 8 [1,30]: In semi-conditional Grammars the cardinality of any context conditional set is no more than 1. G = (V, T, P, S) be a context conditional grammar G is called a semi conditional grammar if it satisfies the following property. For every $(A \to x, Per, For) \in P$ satisfies $card(Per) \leq 1 \; and \; card(For) \leq 1$.

1.3.2 Rule based regulation

While previously we have discussed the regulation which was based on the context of the grammar. In this, we apply regulations on how the grammar rules are being used. They are of four types.

Def 9 [1]: Regular controlled languages put a restriction on the use of rules with regular languages. A regular controlled grammar is a pair $H = (G, \Xi)$,where $G = (N, T, \Psi, P, S)$ is a context free grammar which is named as core grammar, $\Xi \subseteq \Psi^*$ is a regular language, called control language.

Def 10 [1]: Matrix Grammar is a special case of regular controlled grammar where a control language is a form of the finite language. H is a context free grammar G extended by finite set of sequences of its rules, which are referred as matrices. It selects the matrix applies the derivation of all of its rules one by one until it reaches the very last rules. It can now complete its derivation or selects another matrix and repeats the derivation in the same way.

Def 11 [1]: A programmed grammar is a quintuple $G = (N, T, \Psi, P, S)$. Where N, T, Ψ , and S are defined as in a context free grammar. $P \subseteq \Psi \times \mathrm{N} \times (\mathrm{N} \cup \mathrm{T})^* \times 2^{\Psi} \times 2^{\Psi}$ is a finite relation which is called as the set of rules such that $card(\Psi) = card(P)$.

Def 12 [1]: A state grammar is a quintuple $G = (V, W, T, P, S)$. Where, V is a total alphabet, W is a finite set of states, $T \subset V$ is an alphabet of terminals, $S \in V - T$ is the start symbol, $P \subseteq \big(W \times (V - T)\big) \times (W \times V^+)$ is a finite relation.

1.4 Fuzzy sets

Fuzzy sets are those sets which have a degree of membership. In usual set theory, we have two things, either the set is a member, or it's not a member, which is a simple binary relation. In fuzzy sets, the membership can take a real value ranging from [0,1]. This value is defined by membership function which takes certain parameters and then produces a membership value of a certain string. This type of membership is very useful where we may not have all the information, or the information is not very

accurate. Let Σ is a set of alphabets which have a finite set of terminals. ε is empty input. The language $L(G)$ is generated the grammar G. $\mu L(\alpha)$ is the degree of membership for the input α for L. Replacements are denoted by $-$ Operator which is used in place of any symbol from Σ.

Fuzzy self-regulated automata are self-regulated automata which also uses the concept of fuzziness for its acceptance of certain strings which may not be fully accepted by this self-regulated automaton, but they accept them up to a certain degree of acceptance [2,3]. A fuzzy self-regulating finite automaton, SFA, M_{SF} is defined as $M_{SF} = (Q, \Sigma, \delta, q_0, q_t, F, R, E, \mu)$. Where $(Q, \Sigma, \delta, q_0, q_t, F, R)$ these are usual SFA tuples and E is the error set of productions $\mu: P \rightarrow [0,1]$ is the membership function.

Errors are classified into three types [2-3,15]:

- **Replacement Error**: A terminal is replaced with another terminal form Σ. The membership for this kind of operation is found to be 0.6 which is calculated based on different input dataset.
- **Addition Error**: A terminal is placed before or after another terminal form Σ. The membership for this kind of operation is found to be 0.3 which is calculated based on different input dataset.
- **Removal Error**: A terminal is simply removed from the production. The membership for this kind of operation is found to be 0.6 which is calculated based on different input dataset.

We have also tried to apply the concept of fuzziness n-parallel right linear grammar. This uses the production rules and tries to apply the fuzziness on them. The fuzzy rules usually have a degree of membership. Usually the rules accept a certain string with only a degree of 1 when we apply the concept of fuzziness on them we try to accept a string with a certain degree of membership. The degree of membership of for each production depends upon the kind of error it is. This degree of membership varies between 0 and 1. Now we can show that we can also accept an erroneous string with a certain degree of membership that would not have been accepted.

Chapter 2
Literature Survey

2.1 Related Work

In this chapter, we will discuss the work done by researchers on self-regulating finite automata and fuzzy n-parallel right linear grammar. We will try to list the amount of work that has been done by previous researchers and scientists in this field. There have been so many previous works which look forward to enhance the research. During the initial proposition of different kind of automata, different automata models were proposed. Different methods have different specialities and were assigned a specific task to complete. Initially, there were regular grammars which were able to perform very basic acceptance mechanism. But this was not enough for the practical purposes, and the power of this grammar was very limited.

With the evolution of the grammars, there was also a need for the certain cases where the context was required. In some of the cases context was considered and in some context was not considered, the one in which context was considered is called sensitive, and the which did not consider is called as context free grammar. Context free grammar was used when we replace the symbol but do not consider the context of that symbol and simply replace it by the new symbol without worrying about the neighbouring symbols.

2.2 Types of Regulated Automata

The selection of rules chosen for current reduction depends on the rule which was made previously in the previous reduction. It applies for both finite and pushdown automata. Both finite and pushdown version of this automata exist in the regulated form, and they have different properties and powers which depend on the automata. They are self-regulating finite and pushdown automata. However, many properties of pushdown automata which are analogous to its finite version is still an open question.

2.2.1 Self-regulating finite automata

Self-regulating finite automata [1-2] regulate the selection of rules for the current move which depends on the previous move which was made earlier. Let the automata be M, and there exists a binary relation R over the set of rules which are defined in M. Let’s say a sequence of the move is required for the reduction of a certain string. For

that, we have a set of consecutive subsequences which are defined by s_1. $S = s_1 s_2 s_3 s_4 \dots s_n$ Here these set of subsequences form the string S which was required. $|s_i| = |S_j|\ 0 \le i, j \le n$ in which r_j^i denotes the rule according to which the ith move in s_j is made, for all $0 \le j \le n\ and\ 1 \le i \le |s_j|$. If for all $0 \le j < n, (r_1^j, r_1^{J+1}) \in R$,Then M is called as n turn first move self regulating finite automata with respect to R. If for all $0 \le j < n\ and\ all\ 1 \le i \le |s_i| (r_i^j, r_i^{J+1}) \in R$ then it is called as n turn all move self-regulating finite automata [1-2, 25].

Here it gives rise to an infinite hierarchy of language families which coincide with other language families depending on its nature. n-turn all move self-regulating automata gives rise to language family which is generates (n+1) right linear simple matric grammar. Which also concludes that n-turn all move self-regulating automata accepts a proper language which is accepted by (n+1) turn all move self-regulating finite automata. [1,2,28]. A self-regulating finite automata is a septuple [1] $M = (Q, \Sigma, \delta, q_0, q_t, F, R)$ where, $(Q, \Sigma, \delta, q_0, \text{F})$ is a finite automaton, $q_t \in Q$ is a turn state and $R \subseteq \Psi \times \Psi$ is a finite relation on the alphabet of rule labels. Let $\psi: \delta \to \Psi$ be a bijection (Ψ is a set of rule labels).

$$r.qw \to p\ means\ \psi(qw \to p) = r$$

$$qwy \Rightarrow py[r] if\ qwy \in Q\Sigma^*, r.qw \to p \in \delta$$

$$L(M) = \{w \in \Sigma^* : q_0 w \Rightarrow^* f, f \in F\}$$

Def 14 [1] A n-first self-regulating finite automaton is a type of self-regulating finite automata which following properties. It has seven tuples $M = (Q, \Sigma, \delta, q_0, q_t, F, R)$ for $n \ge 0$, Where $(Q, \Sigma, \delta, q_0, F)$ is a finite automaton, $q_t \in Q$ is a turn state, $R \subseteq \Psi \times \Psi$ is a finite relation on the alphabet of rule labels.

Example 2.1 Consider 1-first SFA which is defined as $M = (\{p, q, r\}, \{x, y\}, \delta, p, q, \{r\}, \{1,3\})$ with δ containing rules

1.$px \to p$

2.$px \to q$

3.$qy \to r$

4.$ry \to r$

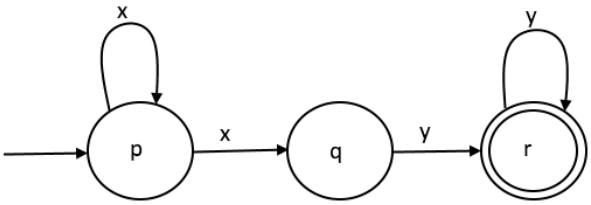


Fig 2.1: n-first self-regulating finite automata [30]

After all of the above reduction process, we can see that the string is accepted. We can see that the language is $L = \{x^n y^n | n \geq 1\}$ which belongs in the category of Context free – Regular Language.

Def 15 [7] N-parallel right linear grammar is a type of parallel grammar which has following properties. An n-PRLG, n>0 is an (n+3) tuple $G = (N_1 \ldots . N_n, T, S, P)$. Where N_i are mutually disjoint non terminal alphabets, $1 \leq i \leq n$. T is a terminal alphabet, $S \notin N_1 \cup \ldots \cup N_n$, P contains three kinds of rules:

$S \to X_1 \ldots X_n \quad X_i \in N_i, 1 \leq i \leq n,$

$X \to wY \qquad X, Y \in N_i \; for \; some \; i, 1 \leq i \leq n, w \in T^*$

$X \to w \qquad X \in N, w \in T^*$

Example 2.2 Let $G = (\{P\}, \{Q\}, \{x, y\}, S, P)$ is a 2-PRLG , where P contains the following rules

$$S \to PQ$$
$$P \to xP$$
$$Q \to yQ$$
$$P \to x$$
$$Q \to y$$

The derivation steps are shown below

$$S \to PQ \to xPyQ \to xxPyyQ \to x^3y^3$$
$$L = \{x^n y^n : n \geq 1\}$$

Def 16 [5] N-Right linear simple matrix grammar has (n+3) tuple, $G = (N_1, \ldots, N_n, T, S, P)$ WhereN_i are mutually a disjoint nonterminal alphabet, $1 \leq i \leq n$. T is a terminal alphabet, $S \notin N_1, \ldots, N_n$, P contains three kinds of matrix rules:

$[S \to X_1, \ldots, X_n] \; X_i \in N_i, 1 \leq i \leq n$

$[X \to w_1 Y_1, \ldots, X_n \to w_n Y_n] \; w_i \in T^*, X_i, Y_i \in N_i, 1 \leq i \leq n$

$[X_1 \to w_1, \ldots, X_n \to w_n] X_i \in N_i, w_i \in T^*, 1 \leq i \leq n$

Example 2.3

Let G=({P}, {Q}, {x, y}, S, P) be a 2-PRLG where P contains rules

$$S \to PQ$$
$$P \to xP, Q \to xQ$$
$$P \to yP, Q \to yQ$$
$$P \to \varepsilon, Q \to \varepsilon$$

Consider the derivation $S \to PQ \to xPxQ \to xyPxyQ \to xyxPxyxQ \to xyxxyx$. The language is $L(G) = \{ww: w \in \{x, y\}^*\}$.

Def 17 [1] N-turn all move self-regulation finite automaton is a type of self-regulating finite automata. It has 7 tuples $n \geq 0$, M is an n-first SFA, and M accepts w if there is $q_0 w \Rightarrow^* f[\mu], f \in F, s.t.$ and $(r_i^j, r_i^{j+1}) \in R$ For all $1 \leq i \leq k, 0 \leq j < n$.

Example 2.4

Consider the example $M = (\{p, q, r\}, \{x, y\}, \delta, p, q, \{r\}, \{(1,4), (2,5), (3,6)\})$ Where the rules are

$$px \to p$$
$$py \to p$$
$$p \to q$$
$$qx \to q$$
$$qy \to q$$
$$q \to r$$

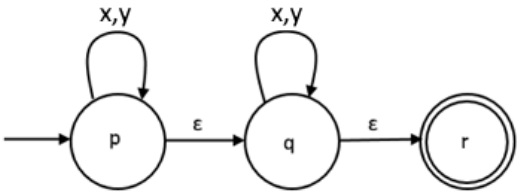


Fig 2.2: 1-turn all self-regulating finite automata [30]

The language generated by this grammar is $L(M) = \{ww: w \in \{a, b\}^*\}$ and this language belongs to the family of Context sensitive – Context free. We can see that the above language "ww" is actually context sensitive. As we can see that by applying regulation on the rules we can clearly achieve the objective by using context free rules and applying regulation on it. Now we don't need context sensitive rules for this grammar.

Meduna and Zemek [1] proposed the concept of regulated automata; later he also added the properties of this kind of automaton, their uses, their types and their various applications. The categorization of these automata is based on different constraints that are applied on automaton to make it a regulated automaton. He also proposed different types of regulated grammars, which are also formed on the basis of types of constraints that are put on the production rules. These various types of grammars exploit various properties of certain grammars and modifying them and editing them to make them work. This regulation of automata which is put on the grammars enable so many capabilities of grammars which were previously not available. They were available, but it was available through a very tedious way using complex things, in this case, may be certain types of context free grammar.

In this particular case of regulated automata, he has shown how we can simply get the job of context sensitive grammar done by using context free grammar by applying the regulation in the context free rules. This has eased the job of many applications where it was firstly mandatory to write a context sensitive grammar which was a complex task. The regulated automata have been useful in many applications like in the field of bioinformatics, natural language processing, compiler deign and many other places where it can be useful. He has also worked on the development of retargetable decompiler which is used for virus detection, and this is independent of any platform. He worked on formal language and helped in the formulation of many languages which can be expressed in the mathematical form.

Kalra and Kumar [2] applied the concept of fuzziness in state grammar. In this paper, they have applied the concept of fuzziness of state grammars. Later this paper explores how it can be applied to the deep pushdown automaton. This concept can be applied to various types of grammars and automaton and increase their capabilities further. It helped me understand how we can accept even the erroneous strings which were not acceptable previously using the standard approach. It shows they can also be accepted up to a certain degree of membership.

Roseburgh [5] explored the concept of right linear simple matrix grammars and later the concept of parallel right linear grammar which is a special case of linear simple matrix grammars. He has shown various examples of these grammars and explores how it can be applied to various aspect of theoretical computer science. It depicts

various properties of n parallel right linear grammar and thus exploits their peculiarities.

Dassow and Paun [8] tries to explain so many different ways of regulation that can be applied over context free grammars and tries to overcome the limitation posed by context free grammars. It tries to regulate the grammar by many different regulating mechanisms. It shows the hierarchy of such regulated grammars, their different properties in the area of decidability. Some mechanism is also shown where the author tries to mix different methods to create a new kind of regulation.

Asveld [3] applied the concept of fuzziness in context free grammars where it tries to accept the erroneous strings which can be used in parsing and various applications. This also helps in determining the level of error we ae doing whether it is a big error or it is a small error. It shows a method to recognize fuzzy languages. It shows a mathematical structure of the fuzzy language. It then shows the closure properties of these fuzzy languages.

Revesz [18] encloses various concepts of so many types of formal languages and shows various operations on them and their result and the variations that will occur when they are applied to them. It explains their decidability and analysis on their structure. It shows how various real life events can be used to by these formal languages to solve these real life models.

Table 2.1 Literature Summary

Name of the Author	Main Features and Key Findings
Meduna and Zemek[1]	i. Introduced the concept of regulated automata. ii. They have shown different types of regulation methods that can be applied to regulate the grammars. iii. Acceptance power of this regulated automaton.
Kalra and Kumar[2]	i. They have applied the concept of fuzziness on state grammars. ii. Fuzziness is further applied on deep pushdown automata.

Roseburgh [5]	i. Explored right linear simple matrix grammar and right linear parallel grammars. ii. Shows various closure properties of this grammar and thus also represents their acceptance as well.
Dassow and Paun[8]	i. It shows different regulation methods that can be applied over context free grammars. ii. Further, explores the properties of these regulated grammars.
Asveld [3]	i. It has applied the concept of fuzziness on the context free grammars. ii. He has also worked in recognition of these fuzzy languages. iii. Also, various properties of these fuzzy languages are shown.
Dallas and Wynn[17]	i. Discussed methods of the improvement of the business process model, tuning its various parameters to improve the efficiency and reduce the total cost of that business model.
Resev [18]	i. Covers the concept of many formal languages, discusses their real life applications. ii. Discusses the complexity of these languages, syntactic analysis and derivation languages.

The following table concludes several types of regulated grammars that have been introduced by several authors. It also briefly discusses few properties of these grammars. The classification of these grammars is based on their type whether they are rules based or context based. They are also further categorized based on more specific properties.

Table 2.2 Types of Regulated Grammar

Author	Name of the grammar	Details
Mitrana and Vide[35]	Context-Conditional Grammars	• This is based on context free rules; each rule can use many strings for its extension up to a finite level. • These extensions may use some permitting strings or can use some forbidding strings. • These rules can rewrite the sentential form but there is one condition that all the permitted strings should appear in it and all the forbidden should be omitted from it.
Dassow and Paun [36,37]	Random Context Grammars	• Random context grammar is a special case of context conditional grammars, where the degree of random context grammars is (1,1).
Dassow and Paun [36,37]	Generalized Forbidding Grammars	• This is the generalized version of forbidding grammars whose degree is (0,1). • The conditions used by this grammar for forbidding is formed using finite languages.
Masopust [38]	Semi-Conditional Grammars	• Semi conditional grammars are those grammars where conditional cardinality does not exceed 1.
Masopust [38]	Simple Semi-Conditional	• This is same as semi conditional grammar but the restriction is that

	Grammars	every rule has no more one condition.
Masopust [38]	Scattered Context Grammars	• In previous context regulated grammars, we have only applied one rule at each derivation steps. • In this grammar, we can apply more than one derivation step at each step. • It can replace more than one non-terminals in single derivation step.
Mitrana and Vide [35]	Regular-Controlled Grammars	• It is an extension of context free rules which uses regular control language for its extension. • Control string is a sequence of rules.
Dassow and Paun [36,37]	Matrix Grammars	• In this, the extension by control language is a finite set of rules, which is referred as matrices.
Dassow and Paun [36,37]	Programmed Grammars	• In every rule has two sets attached to it. These two are subsets of all set of rules. • Now, these sets are used during the derivation rather than complete and which set is to be replaced depends on the condition.
Kasai [39]	State Grammars	• It is an extension over CFG, but unlike previous grammars, it uses an additional state mechanism. • The leftmost occurrence of non-terminal is rewritten under the current state and after that, state is changed.

The below table summarises various types of regulated automata. These automata are basically when we try to apply regulations on finite automata or pushdown automata. It also has a case where we try to regulate a certain kind of automata which is regulated by some control language. This also can be further classified based on which type automata is being used and which type of control language is used.

Table 2.3 Types of Regulated Automata

Author	Name of the automata	Details
Rosebrugh and Wood[40] Meduna and Zemek[1]	Self-Regulating Finite Automata	• Extension of finite automata. • If one rules if applied, then another one will also be applied after the turn state. • It can be of n-turn first move, or it can be n-turn all move.
Rosebrugh and Wood[40] Meduna and Zemek[1]	Self-Regulating Pushdown Automata	• It is the self-regulated version of pushdown automata. • Zero turn all-move pushdown automata are same as context free grammar, and one turn all move pushdown automata is same as recursively enumerable languages.
Wood [41,42]	Regular-Controlled Finite Automata	• This is an automaton in which there is a control language, in this case, it is a regular language. Now this language controls the use of rules over this automaton. • These regulations can of two types: state controlled or transition controlled. They can be transformed into each other.

Wood [41,42]	Context-Free-Controlled Automata	• It is same as regular controlled finite automata, but the difference is that the control language is a context free language rather than regular language. They do not accept any non CFL.
Wood [41,42]	Program-Controlled Finite Automata.	• Here the control language which is generated by programmed grammars.
Wood [41,42]	Regular-Controlled Pushdown Automata	• If control language is regular, it does not affect the power of pushdown automata.
Wood [41,42]	Linear-Controlled Pushdown Automata	• If control language is linear controlled, then it is more powerful than normal pushdown automata.
Wood [41,42]	One-Turn Linear-Controlled Pushdown Automata	• They are less powerful than normal pushdown automata. It is one turn if it does not make any more than one turn during its pushdown.

Chapter 3
Problem Statement and Proposed Solution

In this chapter, the gaps which exist in the present work are discussed. The problem proclamation, objectives to be accomplished and methodology followed is discussed in this chapter.

3.1 Gap Analysis

Although different languages and grammar exist for different kind of purpose, there still exists the part where certain existing tool cannot be applied because of some constraints. There exist context sensitive languages for proposed problems, but they are too complex and cumbersome to implement which leads to so many difficulties. To overcome these problems, we use the concept of regulated automata to apply regulation for such applications and reduce the effort for that application. To increase the acceptance domain of n-parallel right linear grammar.

3.2 Objectives

The aim of our model automata is to show that many context sensitive applications can be implemented using regulated automata and we have shown such an example. Various objectives are mentioned below:

i. To study various types of regulated grammar and automata.
ii. To propose the concept of fuzzy parallel regulated grammar.
iii. To apply the concept of parallel regulated grammar.

3.3 Proposed Solution

Use the techniques of grammar regulation in the context free grammar and generate regulated automata for that particular grammar. Using fuzziness on parallel grammar.

3.3.1 Methodology

The concept of fuzziness is used in the n-parallel right linear grammar, and different rules were introduced using the concept of fuzziness. Now, these new rules will be used along with the other rules to accept the strings. Regulated automata and grammar are introduced for the Chinese tones.

Chapter 4
Proposed Algorithm

In this chapter, we have applied the concept of fuzziness on the grammar and then try to generate various kind of strings from it using these productions. These fuzzy productions can also be used to accept various strings which were previously not accepted; now they can be accepted with a degree of membership.

4.1 Applying Fuzziness on the Grammar

Under normal circumstances, the grammar accepts certain strings which can be generated by using its productions. Either they are completely accepted or completely rejected. There is no mid-way. Fuzziness provides the solution for it. With using fuzziness, it can accept strings which were previously not accepted up to a certain degree of membership. Those strings which are accepted completely have degree of membership 1. Those which are not accepted have degree of membership 0 and other erroneous strings have a degree of membership between 0 and 1.

Example 1

Consider parallel right linear grammar of $\{a^n b^n c^n \mid n \geq 1\}$ [1]

$S \rightarrow ABC$ (1)

$A \rightarrow aA$ (2)

$B \rightarrow bB$ (3)

$C \rightarrow cC$ (4)

$A \rightarrow a$ (5)

$B \rightarrow b$ (6)

$C \rightarrow c$ (7)

Parallel production rules are $(2, 3, 4)$ and $(5, 6, 7)$

$$P_1 : \begin{cases} A \xrightarrow{1} aA \\ B \xrightarrow{1} bB \\ C \xrightarrow{1} bC \end{cases} \qquad P_2 : \begin{cases} A \xrightarrow{1} a \\ B \xrightarrow{1} b \\ C \xrightarrow{1} c \end{cases}$$

Replacement error:

Here $-$ denotes that any symbol from alphabet can be replaced in place of $-$.

$$A \xrightarrow{0.6} -A \qquad (2)$$

$$B \xrightarrow{0.6} -B \qquad (3)$$

$$C \xrightarrow{0.6} -C \qquad (4)$$

$$A \xrightarrow{0.6} - \qquad (5)$$

$$B \xrightarrow{0.6} - \qquad (6)$$

$$C \xrightarrow{0.6} - \qquad (7)$$

Addition error:

$$A \xrightarrow{0.3} -a-A \qquad (2)$$

$$B \xrightarrow{0.3} -b-B \qquad (3)$$

$$C \xrightarrow{0.3} -c-C \qquad (4)$$

$$A \xrightarrow{0.3} -a- \qquad (5)$$

$$B \xrightarrow{0.3} -b- \qquad (6)$$

$$C \xrightarrow{0.3} -c- \qquad (7)$$

Removal error:

$$A \xrightarrow{0.4} A \qquad (2)$$

$$B \xrightarrow{0.4} B \qquad (3)$$

$$C \xrightarrow{0.4} C \qquad (4)$$

$$A \overset{0.4}{\rightarrow} \varepsilon \qquad (5)$$

$$B \overset{0.4}{\rightarrow} \varepsilon \qquad (6)$$

$$C \overset{0.4}{\rightarrow} \varepsilon \qquad (7)$$

In fuzzy parallel right linear grammar, we consider an error in one of the production rules from $(2, 3, 4)$ and $(5, 6, 7)$.

Replacement error:

$$P_3: \begin{cases} A \overset{0.6}{\rightarrow} -A \\ B \overset{1}{\rightarrow} bB \\ C \overset{1}{\rightarrow} bC \end{cases} \quad P_4: \begin{cases} A \overset{1}{\rightarrow} aA \\ B \overset{0.6}{\rightarrow} -B \\ C \overset{1}{\rightarrow} bC \end{cases} \quad P_5: \begin{cases} A \overset{1}{\rightarrow} aA \\ B \overset{1}{\rightarrow} bB \\ C \overset{0.6}{\rightarrow} -C \end{cases}$$

$$P_6: \begin{cases} A \overset{0.6}{\rightarrow} - \\ B \overset{1}{\rightarrow} b \\ C \overset{1}{\rightarrow} c \end{cases} \quad P_7: \begin{cases} A \overset{1}{\rightarrow} a \\ B \overset{0.6}{\rightarrow} - \\ C \overset{1}{\rightarrow} c \end{cases} \quad P_8: \begin{cases} A \overset{1}{\rightarrow} a \\ B \overset{1}{\rightarrow} b \\ C \overset{0.6}{\rightarrow} - \end{cases}$$

Addition error:

$$P_9: \begin{cases} A \overset{0.3}{\rightarrow} -a-A \\ B \overset{1}{\rightarrow} bB \\ C \overset{1}{\rightarrow} cC \end{cases} \quad P_{10}: \begin{cases} A \overset{1}{\rightarrow} aA \\ B \overset{0.3}{\rightarrow} -b-B \\ C \overset{1}{\rightarrow} cC \end{cases} \quad P_{11}: \begin{cases} A \overset{1}{\rightarrow} aA \\ B \overset{1}{\rightarrow} bB \\ C \overset{0.3}{\rightarrow} -c-C \end{cases}$$

$$P_{12}: \begin{cases} A \overset{0.3}{\rightarrow} -a- \\ B \overset{1}{\rightarrow} b \\ C \overset{1}{\rightarrow} c \end{cases} \quad P_{13}: \begin{cases} A \overset{1}{\rightarrow} a \\ B \overset{0.3}{\rightarrow} -b- \\ C \overset{1}{\rightarrow} c \end{cases} \quad P_{14}: \begin{cases} A \overset{1}{\rightarrow} a \\ B \overset{1}{\rightarrow} b \\ C \overset{0.3}{\rightarrow} -c- \end{cases}$$

Removal error:

$$P_{15}: \begin{cases} A \overset{0.4}{\rightarrow} A \\ B \overset{1}{\rightarrow} bB \\ C \overset{1}{\rightarrow} cC \end{cases} \quad P_{16}: \begin{cases} A \overset{1}{\rightarrow} aA \\ B \overset{0.4}{\rightarrow} b \\ C \overset{1}{\rightarrow} cC \end{cases} \quad P_{17}: \begin{cases} A \overset{1}{\rightarrow} aA \\ B \overset{1}{\rightarrow} bB \\ C \overset{0.4}{\rightarrow} cC \end{cases}$$

$$P_{18}:\begin{cases} A \xrightarrow{0.4} - \\ B \xrightarrow{1} b \\ C \xrightarrow{1} c \end{cases} \qquad P_{19}:\begin{cases} A \xrightarrow{0.4} a \\ B \xrightarrow{1} - \\ C \xrightarrow{1} c \end{cases} \qquad P_{20}:\begin{cases} A \xrightarrow{0.4} a \\ B \xrightarrow{1} b \\ C \xrightarrow{1} - \end{cases}$$

We will have the following kind of productions if we replace two rules from the fuzzy set rules.

Replacement error:

$$P_{21}:\begin{cases} A \xrightarrow{0.6} - A \\ B \xrightarrow{0.6} - \\ C \xrightarrow{1} bC \end{cases} \qquad P_{22}:\begin{cases} A \xrightarrow{0.6} - A \\ B \xrightarrow{0.6} - B \\ C \xrightarrow{1} bC \end{cases} \qquad P_{23}:\begin{cases} A \xrightarrow{0.6} - A \\ B \xrightarrow{1} bB \\ C \xrightarrow{0.6} - C \end{cases}$$

$$P_{24}:\begin{cases} A \xrightarrow{0.6} - \\ B \xrightarrow{0.6} - B \\ C \xrightarrow{0.6} - C \end{cases} \qquad P_{25}:\begin{cases} A \xrightarrow{1} a \\ B \xrightarrow{0.6} - \\ C \xrightarrow{0.6} - \end{cases} \qquad P_{26}:\begin{cases} A \xrightarrow{0.6} - \\ B \xrightarrow{1} b \\ C \xrightarrow{0.6} - \end{cases}$$

Addition error:

$$P_{27}:\begin{cases} A \xrightarrow{0.3} - a - A \\ B \xrightarrow{0.3} - b - B \\ C \xrightarrow{1} cC \end{cases} \qquad P_{28}:\begin{cases} A \xrightarrow{0.3} - a - \\ B \xrightarrow{0.3} - b - B \\ C \xrightarrow{1} cC \end{cases} \qquad P_{29}:\begin{cases} A \xrightarrow{0.3} - a - A \\ B \xrightarrow{1} bB \\ C \xrightarrow{0.3} - c - C \end{cases}$$

$$P_{30}:\begin{cases} A \xrightarrow{1} a \\ B \xrightarrow{0.3} - b - B \\ C \xrightarrow{0.3} - c - C \end{cases} \qquad P_{31}:\begin{cases} A \xrightarrow{1} a \\ B \xrightarrow{0.3} - b - \\ C \xrightarrow{0.3} - c - \end{cases} \qquad P_{32}:\begin{cases} A \xrightarrow{0.3} - a - \\ B \xrightarrow{1} b \\ C \xrightarrow{0.3} - c - \end{cases}$$

Removal error:

$$P_{33}:\begin{cases} A \xrightarrow{0.4} A \\ B \xrightarrow{0.4} b \\ C \xrightarrow{1} cC \end{cases} \qquad P_{34}:\begin{cases} A \xrightarrow{0.4} - \\ B \xrightarrow{0.4} b \\ C \xrightarrow{1} cC \end{cases} \qquad P_{35}:\begin{cases} A \xrightarrow{0.4} - \\ B \xrightarrow{1} bB \\ C \xrightarrow{0.4} cC \end{cases}$$

$$P_{36}:\begin{cases} A \xrightarrow{0.4} - \\ B \xrightarrow{0.4} b \\ C \xrightarrow{1} c \end{cases} \qquad P_{37}:\begin{cases} A \xrightarrow{0.4} a \\ B \xrightarrow{0.4} b \\ C \xrightarrow{1} c \end{cases} \qquad P_{38}:\begin{cases} A \xrightarrow{0.4} a \\ B \xrightarrow{1} b \\ C \xrightarrow{0.4} cC \end{cases}$$

Consider $w = aabc$

$$S \underset{1}{\overset{2}{\rightarrow}} ABC \underset{0.3}{\overset{p_9}{\rightarrow}} a - AbBcC \equiv aaaAbBcC \underset{0.4}{\overset{p_{18}}{\rightarrow}} aaabbcc$$

$\mu(w) = 0.3$

If there exists more than one derivation, then it will be determined by using max-min relation.

Algorithm1: Fuzzy parallel linear grammar

Input: Parallel linear grammar (N, T, S, P, RG).

Output: Fuzzy Parallel linear grammar.

$P' \leftarrow \phi$

For each $p \in P$ of the form $\alpha \rightarrow \beta$

If $\beta \in N^+$

No addition of errors is possible in this rule.

Else

If $a \in rhs(p)$

For each $b \in \Sigma \wedge a \neq b$

Add new productions to P' by replacing a by b with membership value 0.6.

//Replacement error

Add new productions to P' by replacing a by ε with membership value 0.4.

//Removal error

Add new productions to P' by replacing a by $\alpha_1 a \alpha_2$ such that $\alpha_1, \alpha_2 \in \{\Sigma \bigcup \varepsilon\}$

$-\{a\}$ with membership value 0.4. **//Addition error**

End of loop

End If

End of loop

Assign membership value 1 to each production.

End of loop

Single error Introduction:

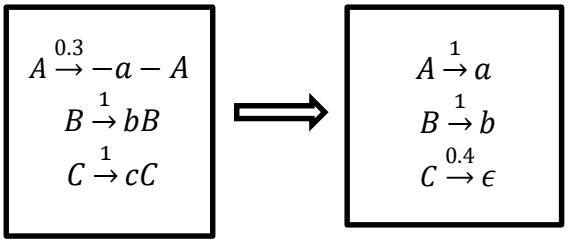
1. For each set of rules to be applied in parallel. Replace one production in each set with a fuzzy one which was derived from the standard production.
2. Now, these are a new set of rules to accept erroneous strings with a degree of membership.

Double error Introduction:

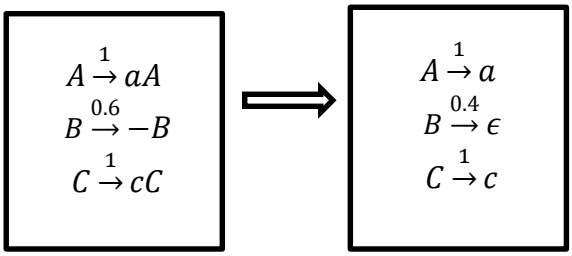
1. For each set of rules to be applied in parallel. Replace two productions in each set with a fuzzy one which was derived from the standard production.
2. Now, these are a new set of rules to accept erroneous strings with a degree of membership.

Consider the derivation strings mentioned below:

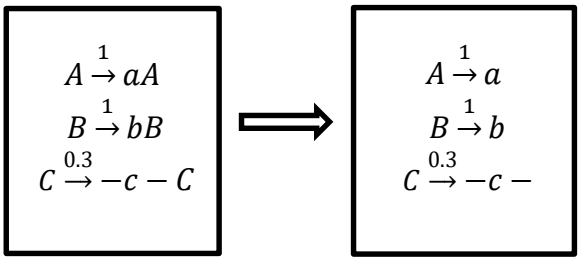
"aaaabbc" its reduction sequence is shown below (Rules in the box are applied in parallel)



So it's membership is 0.3=min(0.3,0.4)

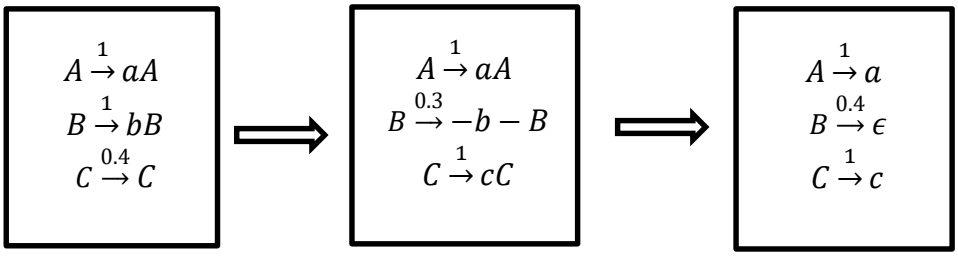


Generated string is "aabcc."
it's membership is 0.4=min(0.6,0.4)



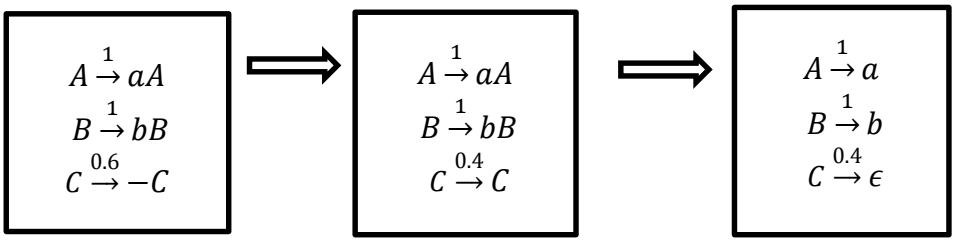
Generated string is "aabbccccccc"

It's membership is 0.3=min (0.3,0.3)



Generated string is "aaabbbbcc"

It's membership is 0.3=min (0.4,0.3,0,4)



Generated string is "aaabbbc"

Its membership is 0.4=min (0.6,0.4,0.4)

To calculate final membership of the string

let $\mu_1, \mu_2 \ldots \mu_n$ be different membership for different reduction paths for the same string where $\mu_x = \min(membership\ of\ all\ reduction\ rules)\ , x = 1,2,..n$

Now final membership μ_F will be calculated as $\mu_F = \max(\mu_1, \mu_2 \ldots \mu_n)$.

Chapter 5
Results

Human possesses a remarkable learning mechanism to acquire new structure using unconscious knowledge. Rohrmeier & Cross [43] classified the structure complexity relevant to implicit learning and found that the inversion operations require supra-finite state complexity. These structures can be classified into various categories based on whether they are processed by finite automata, pushdown automata, linear bounded automata or Turing machine. Parsing complexity of finite automata, pushdown automata and linear bounded automata are linear, polynomial and exponential respectively.

Jiang [44] constructed poems in which last half of sequence of tones was the mirror image of first half of sequence tones and carried out the implicit learning of inversions in a new paradigm. Li, Jiang, Guo, Yang & Dienes [43] repeated the Jiang [44] experiment by using Chinese tonal retrogrades (i. e. central embedding generated context-free grammar) and Inversions (i.e. cross-serial dependencies generated from a mildly context-sensitive grammar). They have pointed out that Chinese tonal retrogrades and inversions can be represented by using memory buffers in the form of last in first out (stack) and first in first out (queue). The corresponding counterpart to represent the inversions operations of Chinese tonal is post machine. The string can be processed in exponential time using the post machine (Post, 1936). The complexity to parse a string using post machine is exponential. In inverse symmetry, elements of a sequence preserve their order, but each element is transformed to its opposite. Fig.5.1 represents the Chinese tones of Tang dynastry poetry in which the second half of the tone sequence is the mirror inversion of the first half of tone sequence [46]. Tonal languages included in many languages in Africa and South East Asia where tones are used to signal different meanings. In Chinese poetry paradigm, four tones are used to signal different meaning. Different syllables meaning are determined according to their tones. For example, the syllable ‘ma’ pronounced in tone1 means mother but horse when pronounced in tone3 [45]. Tone1, tone2, tone3 and tone4 represent flat, rising, falling-rising and falling phonetic characteristics in pitch respectively [46]. The visual esthetic effects of Chinese characters in Tang poetry has been investigated by various researchers [46,48,49,51].

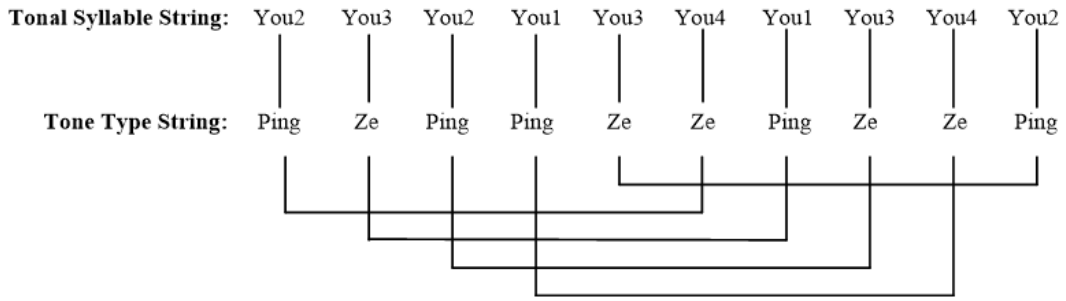


Fig 5.1: Example of grammatical string for inversion. Tone 1 and 2 are categorized as Ping tones, whereas Tone 3 and 4 are categorized a Ze(oblique) tone. Each string consists of 10 tonal syllables, where 1st tone is Ping and 6th tone is Ze, similarly 2nd tone is Ze and 7th tone is Ze and so on.

The concept of self-regulated automata was proposed by Meduna & Masopust [50]. Self-regulated automaton [1] is a regulated automaton which can process the string in polynomial time and can be used to represent a parallel right linear grammar. In this paper, Chinese tonal inversions are represented using self-regulated finite automata and *n*-right linear simple matrix grammar.

5.1 Mathematical Model for representation of Chinese Tonal Inversion Symmetries

Def. 1: A self-regulating finite automata [50] is septuple $M(Q, \sum, \delta, q_0, t, F, R)$ where $(Q, \sum, \delta, q_0, F)$ is a finite state automaton, $t \in Q$ is a turn state, and $R \subseteq \psi \times \psi$ is a finite relation on the alphabet of rule labels.

Def. 2: *n*-turn all-move Self-regulating finite automaton (*n*-all-SFA), $n \geq 0$, is a septuple $(Q, \Sigma, \delta, q_0, t, F, R)$ and M accepts w if there exists $q_0 w \Rightarrow^* f[\mu], f \in F$ such that

$\mu = r_1^0 ... r_k^0 r_1^1 ... r_k^1 ... r_1^n ... r_k^n$ and $(r_i^j, r_i^{j+1}) \in R$ for all $1 \leq i \leq k, 0 \leq j < n.$ [50]

5.2 Self-regulated automata for Chinese Tonal Symmetries

Consider 1-all-SFA $M = (Q, \sum, \delta, t, F, R)$ for representing the inversion operation in Chinese tonal sequence, where

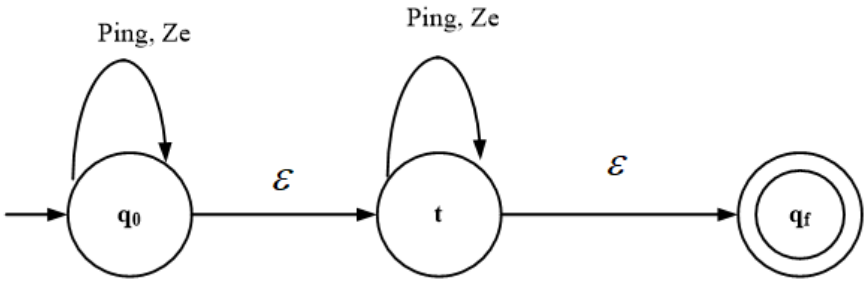


Fig 5.2: Self-Regulated Finite Automata for Chinese Tonal symmetry for Inversion Rules

$$Q = \{q_0, t, f\},$$

$$\sum = \{ping, ze\}$$

$$R = \{(r_1, r_4), (r_2, r_5)\}$$

$$F = \{q_f\}$$

$$r_1 : q_0 \ ping \rightarrow q_0$$

$$r_2 : q_0 \ ze \rightarrow q_0$$

$$r_3 : q_0 \rightarrow q_1$$

$$r_4 : t \ ze \rightarrow t$$

$$r_5 : t \ ping \rightarrow t$$

$$r_6 : t \rightarrow q_f$$

Fig. 5.2 represents the self-regulated automata for Chinese Tonal symmetry.

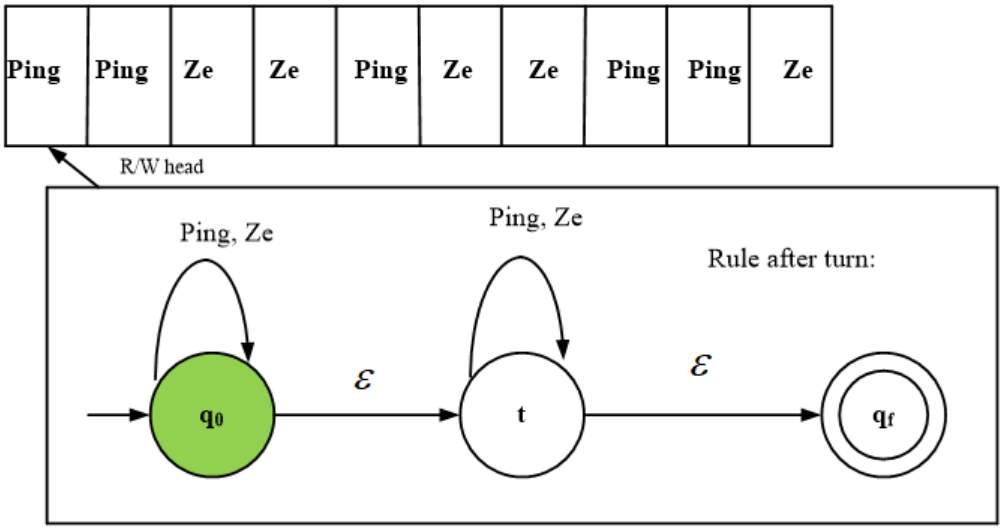


Fig 5.3: Reading 1st symbol Ping Self-regulated finite automata

Consider the input string is Ping Ping Ze Ze Ping Ze Ze Ping Ping Ze. Fig. 5.3 represents the self-regulated automata status while reading the first symbol from the input tape.

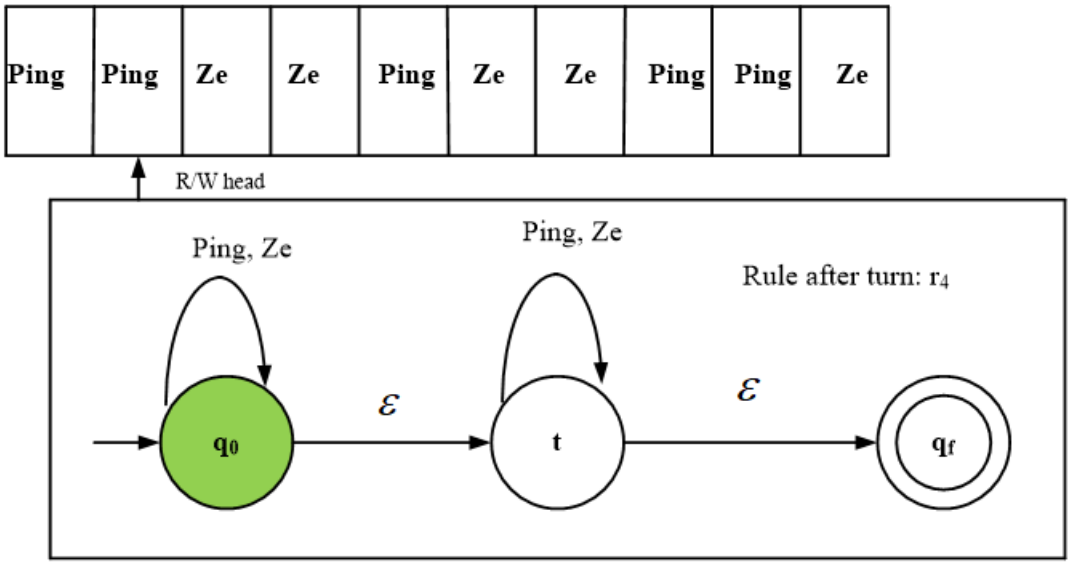


Fig 5.4: Reading 2nd symbol Ping using self-regulate finite automata

The rule $r_1 \in \delta$ is fired and $(r_1, r_4) \in R$ therefore r_4 will be added to the rule after the turn as shown in Fig. 5.4 and Read/write (R/W) head will now point to the second symbol.

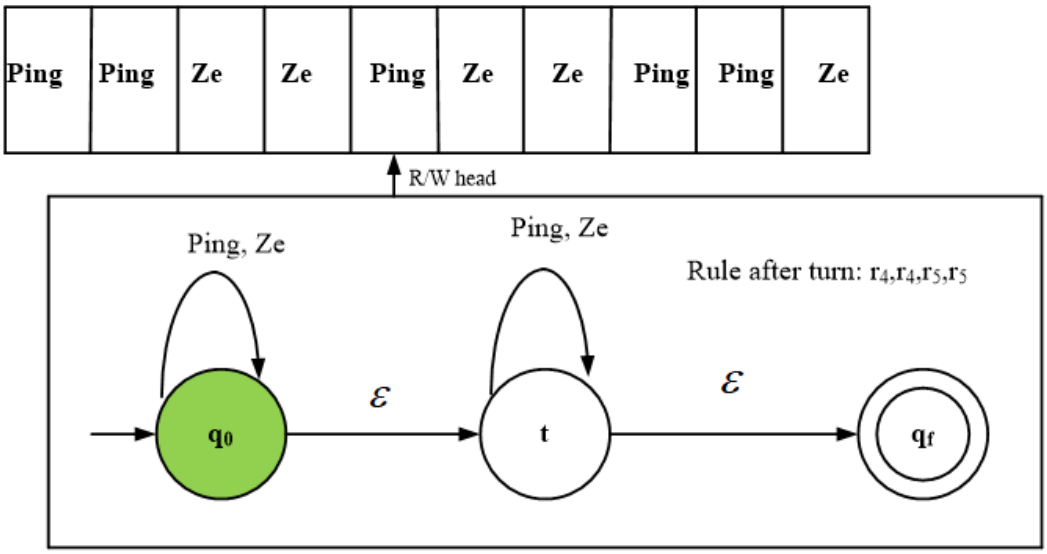


Fig 5.5: Reading Ping after reading 4th symbol using self-regulated finite automata

Above is a representation of the status of the rule after turn state, and self-regulated finite automaton is in the state q_0. After reading 5th symbol, self-regulated automaton moves into turn state t. On reading Ze in the state t, we will apply rule r_4 i.e. the first rule from the rules after the turn.

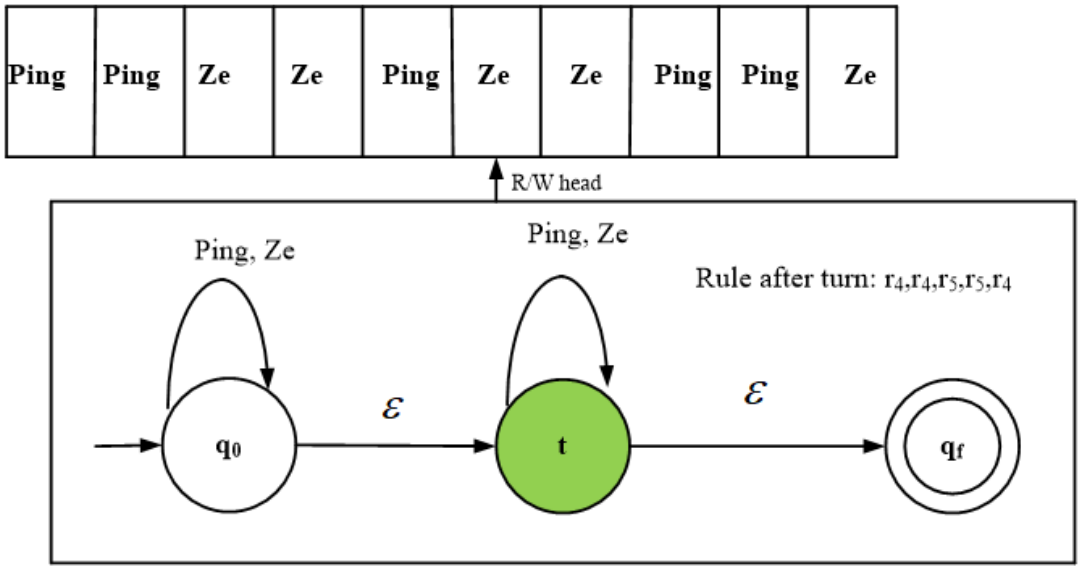


Fig 5.6: Reading 6th symbol Ze using Self-Regulated Finite Automata

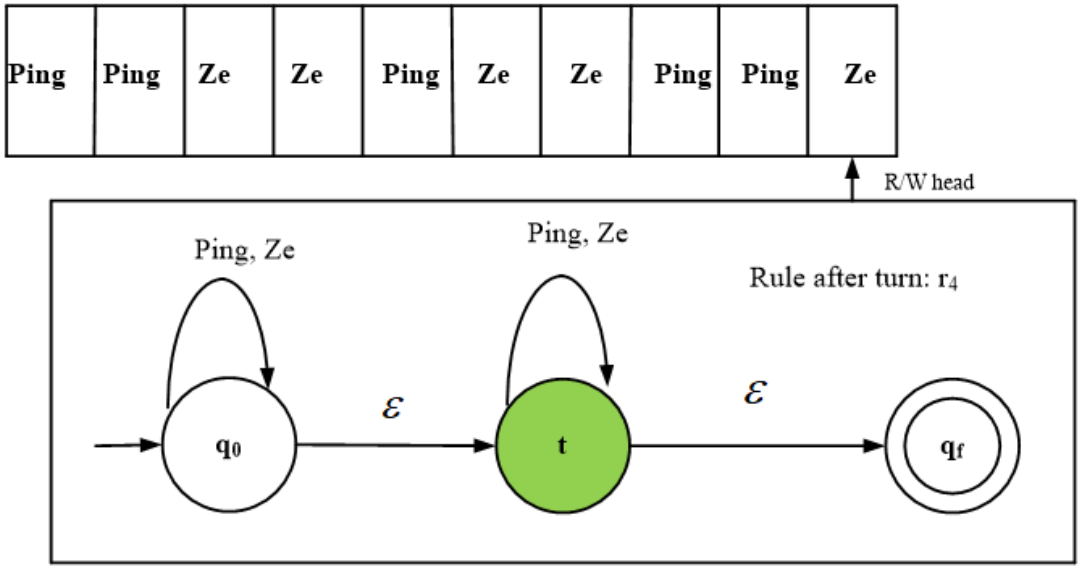


Fig 5.7: Reading last symbol Ping using self-regulated finite automata

Fig.5.7 depicts the status of input tape while reading the last symbol and self-regulated automata will be in the state t.

5.3 n-Right Linear Simple Matrix Grammar for Chinese Tonal Symmetries

Def. 3: An n-Right-Linear Simple Matrix Grammar (n-RLSMG) [1], $n>0$ is an $n+3$ tuples $G=(N_1,...,N_n,T,S,P)$, where

- N_i are mutually disjoint non-terminals, $1\le i\le n$,
- T is a set of terminal symbols,
- $S\notin N_1,N_2,...,N_n$,
- P is a set of production rules of the form:

$$\left[S\to A_1...A_n\right]\,|\,A_i\in N_i\,,1\le i\le n$$

$$[A_1 \to w_1 B_1, ..., A_n \to w_n B_n] \mid w_i \in T^*, A_i, B_i \in N_i$$

$$[A_1 \to w_1, ..., A_n \to w_n] \mid w_i \in T^*, A_i \in N_i$$

n-right linear simple matrix grammar (*n*-RLSMG) for representing the inversion operation in Chinese tonal sequence.

Let $G = (\{A\}, \{B\}, \{Ping, Ze\}, S, P)$, where the production rules are of the form:

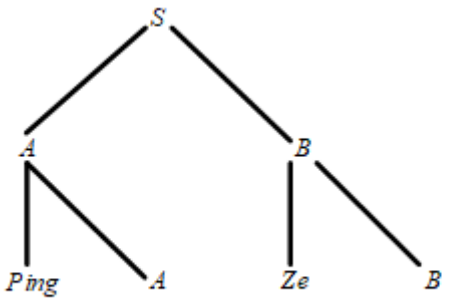
$[S \to AB]$ ……(1)

$[A \to Ping\ A, B \to Ze\ B]$ ……..(2)

$[A \to Ze\ A, B \to Ping\ B]$ …….. (3)

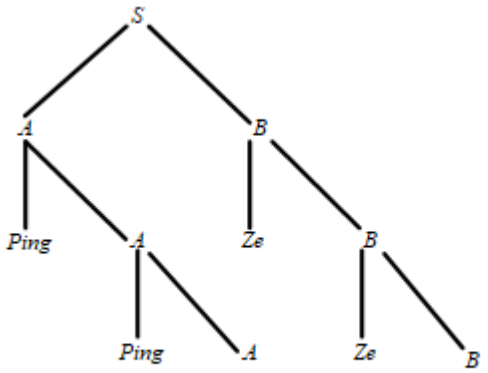
$[A \to \varepsilon, B \to \varepsilon]$ ……..(4)

Consider $w = Ping\ Ping\ Ze\ Ze\ Ping\ Ze\ Ze\ Ping\ Ping\ Ze$



Rule 1

Fig 5.8: Parse tree after applying Rule 1



Rule 1

Rule 2

Fig 5.9: Derivation tree after applying rule 2

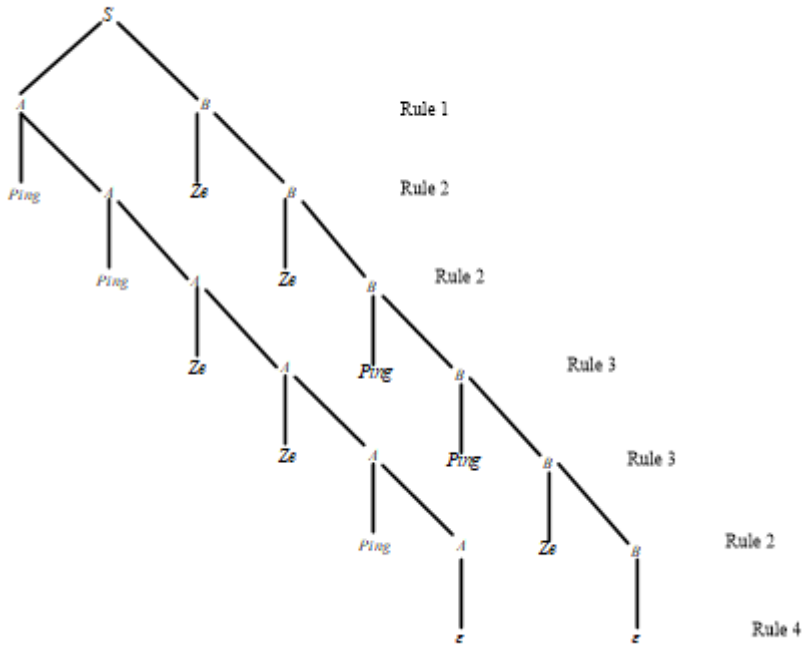


Fig 5.10: Derivation tree for string *w* is represented in.

Our brain unconsciously applies rules similar as *n*-RLSMG (as shown in Fig. 5.8, two transitions for expanding *A* and *B* are applied in parallel).

We have implemented the concept of regulation on an application which uses the context sensitive application. We have constructed a regulated finite automaton for Chinese language tones. Later we have also proposed a right linear simple matrix grammar for Chinese tonal symmetries, and we have also shown the derivation tree for the same. So we have formed a regulated grammar for a process for which it was previously using context sensitive grammars. This has reduced the complexity that was occurring with the context sensitive grammar, and we were also able to define the whole functionality as well. The n-parallel right linear grammars have been used here, and then we have applied fuzziness on those rules. We replaced those rules by the fuzzy version of those rules. These rules were able to accept different strings which were erroneous strings for that grammar. These strings are accepted with a degree of membership. If we have many degrees of membership for a certain string, then we take the maximum of them as the final degree of membership.

Chapter 6
Conclusion and Future Scope

6.1 Conclusion

We have tried to apply the concept of fuzziness of n parallel right linear grammar. In that, we have introduced the concept of fuzziness on the normal productions. What this does is, it makes the rule capable of accepting strings based on a certain degree of membership. Those strings were previously not accepted as they have only 1 degree of production which says that you can either accept this string or completely reject it. There was no midway, but this has opened the way of how we can accept these strings up to a certain extent.

We have also worked on the regulated automata in which we try to put restrictions on the rules and then use these rules as per the restrictions placed on these rules. Further, we have designed an automaton for Chinese tones using the concept of regulated automata.

6.2 Future Scope

The concept of fuzziness can be applied in those areas where we do not have complete information such as in bioinformatics and DNA sequencing and other mathematical calculations. It can be used in those mathematical models in which we have calculations in probabilistic terms, and there is less amount of certainty. Regulated automata can be used in linguistics structure to represent dependencies in the natural language and other aspects of the natural language. It can be very useful in the compiler design and can be used where we have to use context sensitive grammar it can use as an alternative to reduce the complexity of the problem.

References

[1]. A. Meduna and P. Zemek, Self-Regulating Automata, Regulated Grammars and Automata, Springer New York, 509-530, 2014.

[2]. N. Kalra and A. Kumar, Fuzzy state grammar and fuzzy deep pushdown automaton, Journal of Intelligent & Fuzzy Systems, 249-258, 2016.

[3]. R. J. Asveld, Fuzzy context-free languages Part 2: Recognition and parsing algorithms, Theoretical Computer Science, 191–213, 2005.

[4]. M. Schneider, H. Lim and W. Shoaff, The utilization of fuzzy sets in the recognition of imperfect strings, Fuzzy Sets and Systems, 331–337, 1992.

[5]. R. D. Rosebrugh and D. Wood. A characterization theorem for n-parallel right linear languages. J. Comput. System Sci., 579–582, 1973.

[6]. R. D. Rosebrugh and D. Wood. Restricted parallelism and right linear grammars. Util. Math., 151–186, 1975.

[7]. D. Wood. Properties of n-parallel finite state languages. Technical report, McMaster University, 1973.

[8]. J. Dassow and G. Pawn, Regulated rewriting in formal language theory, Springer Publishing Company, 2012.

[9]. J. Dassow, Grammars with regulated rewriting, Formal Languages and Applications, Studies in Fuzziness and Soft Computing 148, Springer Berlin Heidelberg, 249–273, 2004.

[10]. A. Meduna, Formal Languages and Computation: Models and their applications, CRC Press, 2014.

[11]. R.D. Rosebrugh Restricted parallelism and regular grammars M.Sc. Dissertation, McMaster University, 1972

[12]. J. Dassow, Grammars with regulated rewriting, Formal Languages and Applications, Studies in Fuzziness and Soft Computing 148, Springer Berlin Heidelberg, 249–273, 2004.

[13]. A. Meduna, Formal Languages and Computation: Models and their applications, CRC Press, 2014

[14]. Hallerbach, Alena, Thomas Bauer, and Manfred Reichert. "Capturing variability in business process models: The Provop approach." Journal of Software: Evolution and Process, 519-546, 2010.

[15]. J. Dassow and G. Pawn, Regulated rewriting in formal language theory, Springer Publishing Company, 2012.

[16]. P.R.J. Asveld, Controlled Fuzzy parallel rewriting, in: New Trends in Formal Languages—Control, Cooperation, and Combinatorics 1218, Springer-Verlag, 49–70, 1997.

[17]. Ian Dallas, Moe Thandar Wynn,Business Process Management in Small Business: A Case Study Springer 25-46,

[18]. G. E. Revesz. Introduction to Formal Languages. McGraw-Hill, New York, 1983.

[19]. D. Wood. Properties of n-parallel finite state languages. Technical report, McMaster University, 1973.

[20]. G. Paun, A variant of random context grammars: Semiconditional grammars, Theoretical Computer Science 41, 1–17, 1985.

[21]. S. Ginsburg and E.H. Spanier, Control sets on grammars, Mathematical Systems Theory 2(2), 159–177, 1968.

[22]. J.E. Hopcroft, R. Motwani and J.D. Ullman, Introduction to Automata Theory, Languages and Computation, Pearson Education, Singapore, 2000.

[23]. R. Siromoney. Studies in the Mathematical Theory of Grammars and its Applications. PhD thesis, University of Madras, Madras, India, 1969.

[24]. D. Wood. Properties of n-parallel finite state languages. Technical report, McMaster University, 1973.

[25]. M. Ito. Algebraic Theory of Automata and Languages. World Scientific, Singapore, 2004.

[26]. H. R. Lewis and C. H. Papadimitriou. Elements of the Theory of Computation. Prentice-Hall, Englewood Cliffs, 1981.

[27]. J. C. Martin. Introduction to Languages and the Theory of Computation. McGraw-Hill, New York, 1991.

[28]. M. A. Harrison. Introduction to Formal Language Theory. Addison-Wesley, Reading, Massachusetts, 1978.

[29]. A. V. Aho and J. D. Ullman. The Theory of Parsing, Translation and Compiling, Volume I: Parsing. Prentice Hall, Englewood Cliffs, New Jersey, 1972.

[30]. A. Meduna. Automata and Languages: Theory and Applications. Springer, London, 2000.

[31]. R. D. Rosebrugh and D. Wood. Restricted parallelism and right linear grammars. Util. Math., 151–186, 1975.

[32]. T. A. Sudkamp. Languages and Machines. Addison-Wesley, Reading, Massachusetts, 1988.

[33]. Al-Mudimigh, A. S. The role and impact of business process management in enterprise, systems implementation. Business Process Management Journal, 866–874 2007.

[34]. Hammer, M. What is business process management? In J. vom Brocke & M Rosemann (Eds.), Handbook on business process management 1. International handbooks on information systems, Berlin Heidelberg: Springer, 3–16, 2010

[35].Martín-Vide, C., Mitrana, V., P˘aun, G. (eds.): Formal Languages and Applications, chap. 13,pp Springer, Berlin 249–274 2004

[36].Dassow, J., P˘aun, G.: Regulated Rewriting in Formal Language Theory. Springer, New York 1989

[37].P˘aun, G.: A variant of random context grammars: semi-conditional grammars. Theor. Comput. Sci. 41(1) 1985

[38].Masopust, T.: Formal models: regulation and reduction. Ph.D. thesis, Faculty of Information Technology, Brno University of Technology 2007

[39].Kasai, T.: An hierarchy between context-free and context-sensitive languages. J. Comput. Syst. Sci. 4, 492–508 1970

[40].Rosebrugh, R.D., Wood, D.: A characterization theorem for n-parallel right linear languages. J. Comput. Syst. Sci. 7, 579–582 1973

[41].Wood, D.: Properties of n-parallel finite state languages. Technical report, McMaster University 1973

[42].Wood, D.: Theory of Computation: A Primer. Addison-Wesley, Boston 1987

[43].Rohrmeier, M., & Cross, I.: Modelling unsupervised online-learning of artificial grammars: linking implicit and statistical learning. Conscious and Cognition. 27, 155-167, 2014.

[44].Jiang, S.: Implicit learning of Chinese tonal regularity, Ph.D. Thesis, Shanghai, China, East China Normal University, 2012.

[45].Li, F., Jiang, S., Guo, X., Yang, Z. & Dienes, Z.: The nature of the memory buffer in implicit learning: Learning Chinese tonal symmetries. Consciousness and Cognition, 22, 920-930, 2013.

[46].Ling, X., Li, F., Qiao, F. Guo, X., & Dienes, Z.: Fluency Expresses Implicit knowledge of Tonal Symmetry. Frontiers in Psychology, 2016.

[47].Post, E. L.: Finite combinatory processes-formulation. The Journal of Symbolic Logic, 1(03), 103-105, 1936

[48].Chen, W.: Translation of subject ellipsis in Tang Poetry. Journal of Zhejiang University (Humanities and Social Sciences), 36(6), 177-186 (Chinese), 2006.

[49].Liang, L. & Chen, R.: Realization of figure-ground in Tang Poems and its effect on artistic conception. Journal of Foreign Languages (Chinese), 31(4), 31-37, 2008.

[50].Meduna, A. & Masopust T.: Self-Regulating Finite Automata. Acta Cybernetica, 18, 135-153, 2007.

Research Publication

Aatma Vibhor Mishra and Ajay Kumar, "Fuzzy self-regulated Automata", *International Society for Research and Development* – 2017.

[Accepted]

Video Link

The video on thesis can be seen on the following link:

Video Link: https://youtu.be/59fUAAZTgCg