VARIANTS OF ARRAY-REWRITING P SYSTEMS FOR GENERATING PICTURE ARRAYS

PRADEEP ISAWASAN

UNIVERSITI SAINS MALAYSIA
2015

VARIANTS OF ARRAY-REWRITING P SYSTEMS FOR GENERATING PICTURE ARRAYS

by

PRADEEP ISAWASAN

Thesis submitted in fulfilment of the requirements
for the degree of
Doctor of Philosophy

October 2015

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude and heart-felt thanks to the following:

- Dean and Deputy Deans, for giving me a great opportunity to pursue PhD in School of Computer Sciences, Universiti Sains Malaysia;
- Dr Ibrahim Venkat, Supervisor, for his valuable guidance;
- Dr Nurul Hashimah, Co-Supervisor, for supporting me in my progress;
- Prof K.G. Subramanian, for his valuable comments on the direction of my work;
- Academic staff of School of Computer Sciences, Universiti Sains Malaysia for their support;
- Honourable Members of the Examiner panel for their critical comments for improving the thesis, especially at the proposal and research review stages;
- Administration staff of School of Computer Sciences, Universiti Sains Malaysia, especially En Redzuan, Pn Noor Azlina, and Ms Sheela for their kind help;
- My parents, family members and friends for their social and moral support;
- Reviewers for their critical comments in improving the publication related to the thesis;
- Universiti Sains Malaysia for providing an excellent environment to pursue my undergraduate and post-graduate studies;
- Ministry of Higher Education (MoHE), Malaysia for the award of MyPhD scholarship.

I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their helping hand in this venture.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

LIST OF PUBLICATIONS

[1] P. Isawasan, I. Venkat, R. C. Muniyandi, and K. G. Subramanian, "A membrane computing model for generation of picture arrays," in *IVIC15*, vol. 9429 of *LNCS*, Springer, 2015.

[2] K. G. Subramanian, P. Isawasan, I. Venkat, and L. Pan, "Parallel array-rewriting p systems," *Romanian Journal of Science and Technology (ROMJIST)*, vol. 17, no. 1, pp. 103–116, 2014. (ISI 2013 IF: 0.453).

[3] P. Isawasan, I. Venkat, K. G. Subramanian, A. T. Khader, O. Osman, and H. A. Christinal, "Region-based segmentation of hexagonal digital images using membrane computing," in *Asian Conference on Membrane Computing (ACMC14)*, (Karunya Univesity, India), IEEE Explore, 2014.

[4] K. G. Subramanian, P. Isawasan, I. Venkat, L. Pan, and A. Nagar, "Array p systems with permitting features," *Journal of Computational Science (JOCS)*, vol. 5, no. 2, pp. 243–250, 2013. (ISI 2013 IF: 1.567).

VARIAN-VARIAN SISTEM P TATASUSUNAN UNTUK MENJANA TATASUSUNAN IMEJ

ABSTRAK

Bidang pengkomputeran membran dimulakan sekitar tahun 2000, berinspirasikan struktur dan fungsi sel-sel hidup. Model teori pengkomputeran membran ini dipanggil sistem P dan variannya dan penggunaan model ini dalam pelbagai masalah telah disiasat secara intensif sejak itu. Sistem P tatasusunan menghubungkan tatabahasa tatasusunan bahasa formal dengan sistem P. Dalam teori bahasa formal, salah satu kajian utama adalah terhadap keupayaan tatabahasa untuk menjana bahasa, yang disebut sebagai keupayaan generatif, yang bergantung kepada jenis-jenis peraturan yang digunakan. Kami menyiasat keupayaan generatif sistem P tatasusunan dengan memperkenalkan dalam peraturan sistem ciri-ciri benar, tatabahasa dengan penulisan semula selari dan kaedah mengumpul peraturan. Di sini dengan mengaitkan simbol benar dalam kaedah sistem P tatasusunan, kami memperkenalkan varian baru, yang dinamakan sebagai sistem P tatasusunan dengan ciri-ciri benar. Kami membuktikan bahawa jumlah membran yang digunakan dalam pembinaan itu dapat dikurangkan berbanding sistem P tatasusunan. Kami menggabungkan penulisan semula selari dalam sistem P rentetan di dalam sistem P tatasusuan, dengan itu memperkenalkan satu lagi varian baru dalam sistem P tatasusunan dan dinamakan sebagai sistem P tatasusunan selari. Kelebihan sistem baru ini adalah bahawa jumlah membran adalah kecil dalam banyak pembinaan berbanding dengan sebelum ini dalam sistem P tatasusunan. Kami juga menunjukkan

bahawa objek geometri seperti segiempat tepat berongga dan segiempat sama berongga, yang tidak boleh dijana oleh tatabahasa konteks-bebas, dapat dijana oleh sistem P tatasusunan selari diperkenalkan dalam thesis ini. Satu lagi varian baru sistem P tatasusunan, dinamakan sistem P tatasusunan dengan kumpulan peraturan diperkenalkan disini dimana kaedah mengumpul peraturan digunakan sepertimana digunakan di teori bahasa formal. Kaedah ini membolehkan sistem ini menjana corak atau bentuk yang tidak dapat dijana oleh sistem P tatasusunan. Sebagai aplikasi untuk sistem P tatasusunan kami menyiasat sifat struktur iaitu segmentasi rantau daripada tatasusunan imej yang merupakan satu kaedah untuk membezakan rantau dalam sesuatu imej dengan setiap rantau tersebut berkongsi ciri-ciri visual yang sama. Kami menjalankan segmentasi rantau daripada tatasusunan imej heksagon, berdasarkan teknik dalam kes segi empat tepat. Ciri-ciri ketaksaan dalam kejiranan piksel tatasusunan imej heksagon jelas menyumbang kepada pengurangan dalam jumlah peraturan di dalam sistem P. Kaedah segmentasi rantau yand sedia ada berdasarkan sistem P boleh mengendalikan pada satu masa hanya satu saiz tatasususan yang diberikan. Kami memperkenalkan suatu kaedah umum dengan jenis peraturan tatasusunan tertentu dalam sistem P tatasusunan, dengan itu menyediakan satu sistem yang seragam untuk mengendalikan tatasusunan bebas daripada faktor saiz.

VARIANTS OF ARRAY-REWRITING P SYSTEMS FOR GENERATING PICTURE ARRAYS

ABSTRACT

Inspired by the structure and functioning of the living cells, the field of membrane computing was initiated around the year 2000. Since then the theoretical model introduced in this area, called P system has been intensively investigated for properties and applications. One such P system known as array-rewriting P systems provides a link between two dimensional formal language theory and membrane computing. In formal language theory, one of the main studies is on the language generating capability of the grammars, referred to as the generative capacity, which depends on the types of rules. Also a standard technique to increase the generative capacity is to endow the rules with additional features. Here the array-rewriting P system is investigated by endowing the grammatical rules of the system with three such features, namely, permitting symbols, parallel rewriting and grouping of rules. Thus this thesis introduces and develops three such variants of the array-rewriting P system and brings out their advantages. First a new variant, known as permitting array-rewriting P system is introduced, which enables to reduce the number of membranes used in the constructions of P systems in comparison with the original array-rewriting P system. Second variant, called parallel array-rewriting P systems is introduced incorporating the parallel rewriting feature, motivated by parallel rewriting in string-rewriting P systems. Again an advantage of this feature is that the number of membranes is small in many constructions, besides

enabling generation of geometric objects such as hollow rectangles and hollow squares, known to be not generated even by context-free array grammars. A third variant, called tabled parallel array-rewriting P system, is also introduced incorporating the technique of grouping rules into tables of rules. This feature enables to generate picture patterns that cannot be generated by parallel array-rewriting P systems. As an application of the array-rewriting P system model, a structural property, namely region-based segmentation of picture arrays is investigated, which differentiates the regions of a picture array with each region sharing certain visual characteristics. P systems for region-based segmentation of rectangular picture arrays, are known. On the other hand, hexagonal picture array has been of interest in several studies due to its unambiguous connectivity feature. Here extending the technique in the rectangular case, region-based segmentation of hexagonal picture arrays is performed resulting in a reduction in the number of rules. Finally, a general method is proposed with certain specific types of array rewriting rules, thereby providing a uniform framework to handle arrays independent of the size and improving the existing region-based segmentation method.

CHAPTER 1

INTRODUCTION

1.1 Research Background and Motivation

The theory of formal languages came into existence in the mid-1950's with Chomsky [1] proposing grammars in order to model natural language phenomena. Such string grammars serves as language generating devices and thus form the basis of formal language theory. The only relation that exists between adjacent symbols in the one-dimensional strings is that of concatenation. But the role of such one-dimensional structures in applications of computer vision is very restricted [2]. A fundamental approach is to generalize the notion of strings, to higher dimensional structures such as arrays, trees and graphs. Since picture arrays are mainly concerned in the thesis, the extension to two-dimensional arrays is of interest and relevance.

The field of Membrane Computing, initiated by P ăun around the year 2000 [3, 4] has given a new impetus to formal language theory. The computing model of P system [5] in this field, named in honour of its originator, is a computing device which consists of several cell-like membranes placed inside a skin membrane with a hierarchical arrangement and with objects placed in the regions delimited by the membranes. In the basic model, objects are allowed to evolve by evolution rules and can communicate from one region to another, thus leading to a computation of an output. P systems have proved to be a rich theoretical framework to study many computational problems from

wide range of research topics. In fact, in 2013, computer vision has been highlighted in [6] as one of the recently investigated research topics in the framework of P systems. This area includes the problem of generation of picture arrays which has been addressed by researchers by proposing different kinds of grammars. Ceterchi *et al.* [7], in 2003, provided a link between two-dimensional array grammars [8] and membrane computing, introducing array-rewriting P systems, by extending the string-objects P systems to array-objects P systems.

This thesis is dedicated to developing new kinds of array-rewriting P systems with a view to increasing the array generating capacity. In fact, a basic study in formal language theory is on the language generating capability of a class of grammars, referred to as the generative capacity. This depends on the types of rules of the corresponding grammar. For example, rules of a regular grammar are a special form of rules of a context-free grammar. Hence the context-free grammars will have more generative capacity. For example, the language $\{a^n b^n | n \geq 1\}$ cannot be generated by regular grammar rules only as these rules cannot keep track of how many a's are generated in order to match this with an equal number of b's. On the other hand it is well-known that context-free grammar rules can generate this language. In this sense the context-free grammar has more generative capacity than regular grammars. A similar situation prevails in the case of array grammars as well. An application to a problem in the area of computer vision, namely, region-based segmentation of picture arrays, is also dealt with in this thesis.

1.2 Problem Statement

A problem of interest in the array-rewriting P systems is to examine whether the number of membranes can be further reduced in the constructions involved. It is also natural to examine whether this reduction contributes to increase in the generative capacity of the proposed systems. These questions are first addressed in this thesis.

Another problem of interest is to examine the possibility of using array rewriting P systems for region-based segmentation in picture arrays. P systems for region-based segmentation of rectangular digitized picture arrays [9], are known. On the other hand, hexagonal picture arrays have been of interest in investigations in several studies [10]. But a region-based segmentation of hexagonal picture arrays has so far not been attempted. This problem is considered in this thesis as an application of the array P systems. Region-based segmentation of rectangular and hexagonal picture arrays can handle at a time only an array of given size. Hence a problem of interest is to examine whether a more general P system that can handle picture arrays of any size, could be constructed.

1.3 Research Questions

The questions that arise in order to propose solutions to the research problems mentioned above, are listed below:

(1) Array-rewriting P system with permitting features

(a) How does array generation take place in the model of an array-rewriting P

system when it is endowed with permitting feature?

(b) What is the extent to which the generative power can be increased when this feature is incorporated in array-rewriting P system?

(2) Array-rewriting P system with parallel rewriting

(a) What could be the change in the generative capacity of an array-rewriting P system when the rules are rewritten in parallel manner?

(b) How can parallel rewriting incorporated in an array-rewriting P system?

(3) Array-rewriting P systems with table of rules

(a) Will there be any change in the generative capacity when tables of array rewriting rules are incorporated in array-rewriting P system?

(b) How features of tables of rules incorporated in an array-rewriting P system?

(4) Array-rewriting P system for region-based segmentation

(a) How can P systems be utilized to distinguish segments by identifying their borders of different regions in digitized hexagonal arrays?

(b) What will be the difference between region-based segmentation done in rectangular arrays and hexagonal arrays using P system?

(c) What kind of P systems in general can be developed for region-based segmentation that can handle any array size?

1.4 Research Objectives

This thesis focuses on the following objectives listed below:

(1) Investigation of grammatical methods, namely array-rewriting P system for generating digitized picture arrays.

   (a) To *incorporate* in the array-rewriting P system, permitting feature, parallel rewriting and table of array rewriting rules;

   (b) To *define* permitting array-rewriting P system, parallel array-rewriting P system and tabled parallel array-rewriting P system;

   (c) To *examine* the generative capacity of the models introduced.

(2) Investigation of a structural property, namely region-based segmentation of picture arrays.

   (a) To *introduce* array-rewriting P systems that can uniformly handle region-based segmentation of picture arrays of any size;

   (b) To *extend* the methods in the rectangular case to perform region-based segmentation of hexagonal picture arrays;

   (c) To *utilize* the advantage of hexagonal neighbourhood unambiguity in the number of rules.

1.5 Overview of Research Methodology

The research undertaken utilizes and adopts the methods and techniques of two-dimensional grammar theory in the framework of P systems, such as the following:

(1) **Linking array grammars with P systems**

The notion of a P system arising in the context of membrane computing has been found to be a rich theoretical framework for many problems in computing. The thesis employs the notion of a P system in array grammar models.

(2) **Regulating the rewriting**

The technique of regulating the rewriting has been employed successfully in formal language theory. One such method of regulating the rewriting is by allowing permitting symbols, which is used in the study undertaken in the thesis and the effect of this mechanism in relation to the array grammars is explored.

(3) **Parallel application of rules**

The application of the picture grammar rules can be done in parallel, which again is a standard technique used in string grammar theory. This might result in increase in the generative power and this aspect is studied in this thesis.

(4) **Grouping of rules**

The technique mainly used in Lindenmayer systems that is grouping of string rules in string generating systems has been adopted for array-rewriting rules in array rewriting systems. This might enable to generate picture arrays that cannot be generated by other variant of array-rewriting P systems.

(5) **Region-based segmentation**

Investigation of the problem of region-based segmentation by designing a family of P systems that assigns labels to the pixels in a rectangular picture array in such a way that pixels with similar features are assigned the same label, has been done.

This method is adopted for performing region-based segmentation of hexagonal picture arrays.

1.6 Thesis Organisation

In Chapter 2, a literature review is done in order to place the research work in context. This chapter discusses the various types of grammatical methods for generating picture arrays. The chapter also reviews array-rewriting P system and its generative power. In Chapter 3, a permitting array-rewriting P system is introduced by associating permitting symbols with rules in the regions of an array-rewriting P system. The chapter also discusses the advantage of this approach compared to the array-rewriting P system. In Chapter 4, a parallel array-rewriting P system (PAP) is introduced where rewriting of rules are done in parallel. The chapter also discusses the generative power of this model compared with certain array grammars generating array languages. In Chapter 5, a tabled parallel array-rewriting P system (TPAP) is introduced by incorporating in the regions of the PAP, the feature of having tables of rules, well-known in formal language theory and examine the generative power. The generative power of TPAP as well as the ability of this system in describing picture patterns are investigated. In Chapter 6, the concept developed for rectangular arrays is appropriately modified to deal with hexagonal arrays and by using this variant of P system, hexagonal array segmentation is demonstrated. Finally, the contributions of this thesis are summarized and possible future works are outlined in Chapter 7.

CHAPTER 2

LITERATURE REVIEW

In this chapter, discussions on the existing knowledge in terms of substantive findings, as well as theoretical and methodological contributions involved in array-rewriting P systems are done.

2.1 Overview

In the literature three main categories of methods are involved in using grammars for generating picture arrays, whether in the investigation of syntactical methods or structural methods.

(a) Rewriting of rules : Sequential, parallel and hybrid (combination of sequential and parallel) rewriting of rules to achieve desired results.

(b) Creation of rules : A non-isometric variety where array rewriting is a combination of string rules, whereas in isometric variety the rules are created directly analogous to string rules.

(c) Application of rules : Rules could be applied freely without any restriction or the application of rules can be regulated.

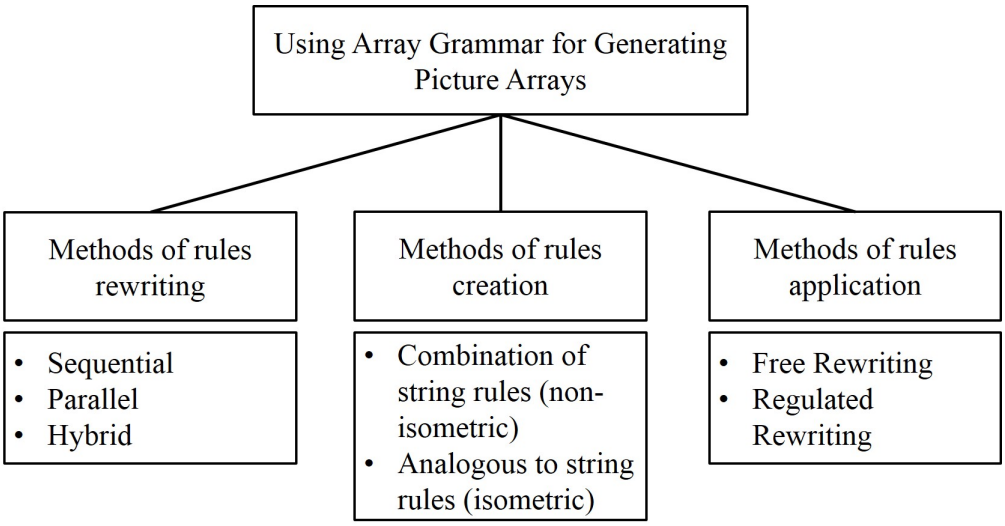

Figure 2.1: Overview of methods involved in using grammars for generating picture arrays

Thus in this chapter discussion on the relevant existing literature on the methods required for picture array generation are done. Here also needed definitions, concepts and statements of results in the theory of formal languages, recalled. Most of the material referred in this thesis can be found in standard textbooks in the area of theoretical computer science (see e.g. [11, 12, 13, 14, 15]).

2.2 Formal Language Theory

What is a language? Dictionary defines the term informally as a system suitable for the expression of certain ideas, facts or concepts including a set of symbols and rules for their manipulation. To study languages mathematically, a mechanism is needed to describe them. In the 1950's Chomsky proposed a mathematical model of a grammar [1] for describing a language.

Since then there has been continued interest and activity among researchers with

various backgrounds both in the theoretical development and in the application areas of formal language theory. This has resulted in a huge amount of established research on the mathematical aspects of formal language theory. Besides this, language theory has contributed to very many diverse fields [11, 12, 13] such as linguistics, information theory, molecular and developmental biology, DNA computing [16, 17], membrane computing [5], pattern matching [18], cryptography [19, 20], computer security [21] and pattern recognition in images [2].

2.2.1 Alphabet and Word

An alphabet V is a finite nonempty set of symbols. A word over an alphabet V is a finite length sequence composed of symbols from the alphabet. The empty word denoted by λ or ε is the word consisting of no symbols. The set of all words over V is denoted by V^* [14]. Examples of common alphabets are e.g. letters in the English alphabet $\{a, b, \ldots, z\}$ and the bits 0 and 1 i.e. $\{0, 1\}$.

Summarization on the basic operations on words are provided in Table 2.1.

Table 2.1: Basic operations on words

Operation	**Written as**	**Representing**	**Examples**
Length	$\|w\|$	The number of symbols in the word $\|w\|$.	$\|aabbcaabc\| = 9$, $\|a\| = 1$, $\|\varepsilon\| = 0$
Concatenation	$w_1 w_2$	The word formed by writing down the word w_1 followed immediately by the word w_2.	Let $w_1 = abbaa$ and $w_2 = cabb$, then $w_1 w_2 = abbaacabb$
Power	w^n	The word formed by writing down n copies of the word w.	Let $w = abbaa$, then $w^3 = abbaaabbaaabbaa$

2.2.2 Grammars

A grammar is a finite device which generates all the words of a language starting from one or more symbols or axioms. It is a rewriting system of a special type where the alphabet is partitioned into two sets of symbols, the so-called terminal symbols (terminals) or constants and the so called nonterminal symbols (nonterminals) or variables, and one of the nonterminals is specified as the axiom.

Definition 2.2.1. [14] A grammar is a quadruple

$$(V, T, S, P)$$

where:

(i) V is a finite nonempty set disjoint from T. The elements of V are called the nonterminals or variables;

(ii) T is a finite nonempty set called the terminal alphabet. The elements of Σ are called the terminals;

(iii) $S \in V$ is a distinguished nonterminal called the start symbol;

(iv) P is a finite set of rules of the form $\alpha \rightarrow \beta$ where α and β are words over $\Sigma \cup V$.

The rules of the grammar (or production rules) are the central of a grammar, they specify how the grammar transforms one string to another and through this they define a language associated with the grammar. In the discussion all production rules are of the form

$$x \rightarrow y$$

where x is an element of $(V \cup T)^+$ and y is in $(V \cup T)^*$ are assumed. The productions are applied in the following manner:

Given a string w of the form

$$w = uxv$$

say the production is applicable to this string and may use it to replace x with y, thereby obtaining a new string

$$z = uyv.$$

This is written as

$$w \Rightarrow z.$$

It can be said that w derives z or that z is derived from w. Successive strings are derived by applying the productions of the grammar in arbitrary order. A production can be used whenever it is applicable, and it can be applied as often as desired. If

$$w_1 \Rightarrow w_2 \Rightarrow \quad \Rightarrow w_n$$

it can be said that w_1 derives w_2 and write

$$w_1 \Rightarrow^* w_n.$$

The * indicates that an unspecified number of steps (including zero) can be taken to derive w_n from w_1.

2.2.3 Language

By applying the production rules in a different order, a given grammar can normally generate many strings. The set of all such terminal strings is the language defined or generated by the grammar.

Definition 2.2.2. [14] Let $G = (V, T, S, P)$ be a grammar. Then the set

$$L(G) = \{w \in T^* : S \Rightarrow^* w\}$$

is the language generated by grammar G.

Example 2.2.1 will illustrate how grammar used as a finite mechanism to model a infinite set. Here in this example a dataset consisting of elements with equal number of a and b arranged with a first followed by b

$$\{ab, aabb, \quad a^n b^n, \quad\}$$

are used or can be formally represented as a language

$$L(G) = \{a^n b^n : n \geq 1\}$$

Example 2.2.1. Consider the grammar

$$G = \{V, T, S, P\}$$

where

(i) $V = \{S\}$;

(ii) $T = \{a, b\}$;

(iv) $P = \{$ Rule 1 : $S \rightarrow aSb$, Rule 2 : $S \rightarrow ab\}$

Then

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaabbb.$$

The word $aaabbb$ is an element in the language generated by the grammar G.

2.3 Chomsky's Hierarchy

The Chomsky hierarchy described by Chomsky in 1956 [1] is a collection of four classes of formal languages, each of which is a proper subset of the classes above it, and each of which corresponds to a generating grammar.

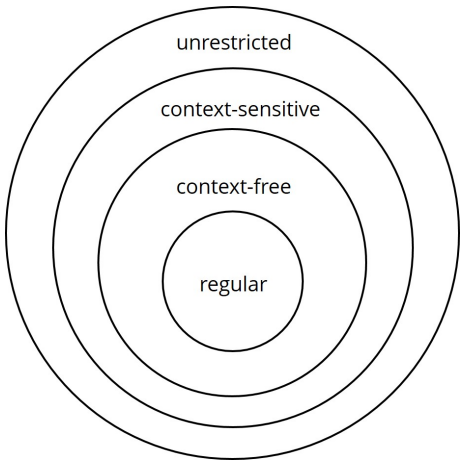

Figure 2.2: Set inclusions described by the Chomsky hierarchy

Each level of this hierarchy consists of a class of formal languages, a class of generative grammars, each of which produces a language in the associated class. At each

level of the hierarchy, the rules for the generative grammar become more restrictive, making each class of languages a subset of the classes above it. Those four classes, from least restrictive to most restrictive, are:

Type 0 - **No restrictions:** the productions are the form $u \rightarrow v$, where $u \in V^+ - \{\lambda\}$, $u \in N$ and $u \in V^*$.

Type 1 - **Context sensitive grammars:** the productions are of the form $u \rightarrow v$, where $u = u_1 A u_2$, $v = u_1 Z u_2$ for $u_1, u_2 \in V^*$, $A \in N$ and $z \in V^*$.

Type 2 - **Context-free grammars:** the productions are of the form $A \rightarrow v$, where $A \in N$ and $v \in V^*$.

Type 3 - **Regular grammars:** the production are of the form $A \rightarrow aB$ or of the form $A \rightarrow Ba$ where $A, B \in N$ and $a \in T$.

For further reading on Chomsky's Hierarchy, refer to [1, 11].

2.4 Regulated Rewriting

It is well-known that context-free grammars cannot cover all aspects of natural languages, programming languages and other related fields. Therefore a lot of mechanisms have been introduced which regulate the application of context-free rules.

The study of regulated rewriting [22, 23] is a significant branch of formal language theory, developed mainly with the aim of increasing the generative power of context-free grammar (to generate as large families of languages as possible using as simple machineries as possible, extensions of context-free grammars).

In a given grammar, rewriting can take place at a step of a derivation by the usage of any applicable rule in any desired place. That is if A is a nonterminal occurring in any sentential form say $\alpha A\beta$, the rules being

$$A \to \mu, A \to \delta$$

then any of these rules is applicable for the occurrence of A in $\alpha A\beta$. Hence one encounters nondeterminism in its application. One way of naturally restricting the nondeterminism is by regulating mechanisms, which can select only certain derivations as correct in such a way that the obtained language has certain useful properties. For example a very simple and natural control on regular rules may yield a non regular language.

Matrix grammar is one kind of regulated rewriting.

Definition 2.4.1. [22, 23] A matrix grammar is a quadruple

$$G = (N, T, S, P)$$

where N, T and S are as in any Chomsky grammar. P is a finite set of sequences of the form:

$$m = [\alpha_1 \to \beta_1, \alpha_2 \to \beta_2, \ldots, \alpha_n \to \beta_n]$$

$n \geq 1$ with $\alpha_i \in (N \cup T)^+$, $\beta_i \in (N \cup T)^*$, $1 \leq i \leq n$, m is a member of P and a matrix of P.

By regulating the rewriting in a grammar the changes in the generative power will

be illustrated in the matrix grammar example below.

Example 2.4.1. Let $G = (N, T, S, P)$ be a matrix grammar with

(i) $N = \{S, A, B, C\}$

(ii) $T = \{a, b, c\}$

(iii) $S \rightarrow ABC$

(iv) $P = \{ m_1 = \begin{matrix} A \rightarrow aA \\ B \rightarrow bB \\ C \rightarrow cC \end{matrix}, m_2 = \begin{matrix} A \rightarrow a \\ B \rightarrow b \\ C \rightarrow c \end{matrix} \}$,

Start the application of the rules with

$$S \Rightarrow ABC$$

When the matrix m_2, the string is used

$$S \Rightarrow ABC \overset{*}{\Rightarrow} abc$$

will be generated and the system halt because the matrix m_2 is a terminating matrix. Instead when the matrix m_1

$$S \Rightarrow ABC \Rightarrow aAbBcC \Rightarrow aaAbbBccC \Rightarrow aaabbbccc$$

is used the string can grow until the termination matrix m_2 is applied. Hence the language generated is

$$L = \{ a^n b^n c^n | n \geq 1 \}.$$

Note that the language generated is a context-sensitive even though the rules used is a context-free. Therefore it managed to generate a language belonging to a higher class, increasing the generative power by putting restriction on the manner of applying the rule.

2.5 Array Grammars

String generative devices such as Chomsky grammars form a central component in the theory of formal languages. Extending these string grammars several methods have been proposed for generating picture arrays which are finite connected arrangements of labelled pixels in the two-dimensional plane. Grammatical methods constitute one of the most popular forms of generating picture arrays. Array grammars [24, 25, 26], Lindenmayer systems [27, 28], chain-code picture grammars [29, 30], collage grammars [31], puzzle grammars [32, 33, 34], pure two-dimensional picture grammars [35], contextual array grammars [36] are some of the grammars introduced in the literature for generating picture arrays.

In the literature, the early devices for picture generation were proposed in the late 1960's and early 1970's, such as the array grammars of Rosenfeld [37, 38, 39], Siromoney *et al.*[40, 41], and the shape grammars of Stiny and Gips [42]. Array grammars are also referred to as picture grammars or two-dimensional grammars [24, 25, 26]. The term array grammar is uniformly used throughout the thesis.

In extending the study of formal (string) languages to two dimensions, there has been a continued effort on the part of the researchers in the area of two dimensional

languages [8, 24, 25, 26] to make use of the techniques of formal string language theory for developing methods to study the problem of two dimensional array generation and description. The literature on grammatical methods for two dimensional array generation and analysis has been steadily growing. Grammatical techniques of generation of digital two dimensional arrays have become established as one of the major areas of theoretical studies in image analysis, basically due to the structure handling ability of the syntactic models. A two dimensional language consists of two dimensional arrays (rectangular or non-rectangular) of symbols.

A number of two dimensional array generating mechanisms with the intention to increase the generative capacity have been introduced in the literature, for example pure two-dimensional (2D) context-free grammar (P2DCFG) [35], basic puzzle grammar (BGP) array P systems [43], extended 2D context-free picture grammar (E2DCFPG) [44], pure 2D context-free picture grammar (P2DCFPG) and P2DCFPG with regular control [45], and P system model with pure context-free rules for picture array generation [46].

2.5.1 Picture Arrays

The picture arrays considered in this thesis consist of finitely many symbols from a specified alphabet V placed in the points of $\mathbf{Z}^2$ (the plane); the points of the plane which are not marked with elements of V are supposed to be marked with the blank symbol $\# \notin V$.

Definition 2.5.1. [22] An array is a mapping $\boldsymbol{A} : \mathbf{Z}^2 \rightarrow V \cup \{\#\}$ with a finite support, $supp(\boldsymbol{A})$, where $supp(\boldsymbol{A}) = \{ v \in \mathbf{Z}^2 \mid \boldsymbol{A}(v) \neq \# \}$.

In order to specify an array, it is sufficient to specify the pixels v of the support, together with their associated symbols from V, hence giving a set of elements of the form

$$(v, \boldsymbol{A}(v)), \quad \text{for } v \in supp(\boldsymbol{A})$$

However, the arrays can be pictorially represented, indicating their non-blank pixels. The examples below illustrates this.

Example 2.5.1. The L-shaped angle with equal arms with length 3 as Figure 2.3 (placed in the center of the plane) is formally given by

$$\{((0,0), a), ((1,0), a), ((2,0), a), ((3,0), a), ((0,1), a), ((0,2), a), ((0,3), a)\},$$

and the assumption is that all other elements of $\mathbf{Z}^2$ contain the symbol #.

a
a
a
a a a a

Figure 2.3: L-shaped array with equal arms with length 3

Only the relative positions of non-blank pixels in the array taken into account.

Example 2.5.2. The T-shaped array with equal arms with length 5 in Figure 2.4, the

specifications is given as follows:

$\{((0,5), a), ((1,5), a), ((2,5), a), ((3,5), a), ((4,5), a), ((5,5), a), ((5,0), a), ((5,1), a), ((5,2), a), ((5,3), a), ((5,4), a), ((5,6), a), ((5,7), a), ((5,8), a), ((5,9), a), ((5,10), a)\}$,

a a a a a a a a a a a
a
a
a
a
a

Figure 2.4: T-shaped array with equal arms with length 5

The leftmost symbol of the upper horizontal arm of T is the pixel $(0,5)$ and the lowermost symbol in the vertical arm is the pixel $(6,0)$. The non-blank labels of the T-shaped array are pictorially indicated in Figure 2.4, since only the relative positions of non-blank pixels in the array really matter for us.

2.5.2 Array Grammars

The array grammars (also called isometric array grammars) [24, 25, 26] involve array rewriting rules that preserve the geometric shape of the rewritten subarray. These are extensions of string grammars [11, 12, 13, 47] to two dimensional picture arrays.

The context-free and regular types of array rewriting rules of the isometric variety

recalled here.

Consider an array grammar

$$G = (V, T, \#, S, P)$$

where

(i) $V = N \cup T$ is an alphabet.

(ii) The elements of the finite set N are called nonterminals and those of T terminals, with $N \cap T = \emptyset$.

(iii) $S \in N$ is the start symbol.

(iv) P is a finite set of array rewriting rules of the form $r : \alpha \rightarrow \beta$, where α and β are arrays over $V \cup \#$ satisfying the following conditions:

(1) The arrays α and β have identical shapes;

(2) At least one square in α is labelled by an element of N ;

(3) The symbols of T that occur in α are retained in their corresponding squares in β;

(4) The application of the rule $r : \alpha \rightarrow \beta$ preserves the connectivity of the rewritten array.

For two arrays μ, δ over V and a rule r as above, write $\mu \Rightarrow_r \delta$ if δ can be obtained by replacing with β, a subarray of μ identical to α. The reflexive and transitive closure of the relation $\Rightarrow$ is denoted by $\Rightarrow^*$.

An array grammar is called:

(1) Context-free if for all the rules

$$r : \alpha \rightarrow \beta$$

the non$^-$ # symbols in α are not replaced by symbol # in β; for each rule $\alpha \rightarrow \beta$, α contains exactly one nonterminal with the remaining squares containing #, and β contains no blank symbol #;

(2) Regular if the rules are of the following forms:

Rule 1 = A # → a B Rule 2 = # A → B a

Rule 3 = $\begin{matrix} \# & \\ A & \end{matrix} \rightarrow \begin{matrix} B \\ a \end{matrix}$ Rule 4 = $\begin{matrix} A & \\ \# & \end{matrix} \rightarrow \begin{matrix} a \\ B \end{matrix}$

Rule 5 = A → B Rule 6 = A → a

where A, B are nonterminals and a is a terminal.

2.5.3 Array Languages

The array language generated by G is

$$L(G) = \{ p \mid S \Rightarrow^* p \in T^{+2} \}$$

Note that the start array is indeed $\{((0, 0), S)\}$ and it is understood that this square labelled S is surrounded by #, denoting empty squares with no labels.

The process of generating a language by an array grammar illustrated with the

following example.

Example 2.5.3. The context-free array grammar G_s with rules

$$r_1 = \begin{matrix} & \# & \\ \# & S & \# \end{matrix} \rightarrow \begin{matrix} & A & \\ D & a & B \\ & C & \end{matrix} \qquad r_2 = \begin{matrix} \# \\ A \end{matrix} \rightarrow \begin{matrix} A \\ a \end{matrix}$$

$$r_3 = \begin{matrix} C \\ \# \end{matrix} \rightarrow \begin{matrix} a \\ C \end{matrix} \qquad r_4 = B\ \# \rightarrow a\ B$$

$$r_5 = \#\ D \rightarrow D\ a \qquad r_6 = A \rightarrow a$$

$$r_7 = B \rightarrow a \qquad r_8 = C \rightarrow a$$

$$r_9 = D \rightarrow a$$

where S, A, B, C, and D are nonterminals and a is a terminal, generates star-shaped arrays with four arms over $\{a\}$.

Computation: In a derivation, starting with S, the rule r_1 is applied once.

$$S \underset{r_1}{\Rightarrow} \begin{matrix} & A & \\ D & a & B \\ & C & \end{matrix}$$

This can then be followed by the application of the rule r_2 as many times as needed, for example let's say 2 times, thus growing the vertical upper arm and the growth can be terminated with an application of the rule r_6.