Introduction

Formal Languages

Born in the middle of 20th century, formal language theory is a tool for modeling and investigating syntax of natural languages. The study on languages and their grammars made by Noam Chomsky during 1950s led to an extensive research in the field of formal languages. After 1964, the theory has evolved as a separate branch with specific problems, techniques and results. Formal languages has been a tool applied to natural language processing with applications in various other fields of research.

Formal language is an abstraction of the general characteristics of programming languages. A formal grammar consists of a set of symbols and rules for the formation by which these symbols can be combined into entities called ‘sentences’. A formal language is the set of all sentences permitted by the rules of formation. The handbook of formal languages edited by Rozenberg and Salomaa [42] gives a comprehensive account of most of the developments that have taken place in the field of formal languages, which is now an area of fundamental importance in theoretical computer science besides having applications in many fields including biology. For basic details on formal languages we refer to [43].
Pattern Languages

Pattern languages were introduced by Angluin [4] while studying the problem of learning or inferring a pattern common to all strings in a given sample. A pattern is a finite string of constants (or terminal symbols) and variables (or non-terminal symbols). A pattern language is the set of all strings obtained by substituting for each variable symbol in the pattern, strings of constants, with different occurrences of the same variable being replaced by the same string. An overview of pattern languages can be seen in [36].

Following the study of Angluin [4], Castiglione et al. [11] studied patterns in words and languages and a new generative device called a pattern grammar had been defined [16, 28]. The idea delineated here is to start from a finite set \( A \) of axioms, which are over an alphabet of constants, given a set \( P \) of patterns which are strings over constants and variables, replace the variables in a given pattern by axioms and continue the process with the current set of strings, obtained by such operations. The replacement in this process is parallel which means that all variables occurring in a pattern can be replaced simultaneously. It is also uniform which means same variables are replaced by same string at a particular step. All strings generated in this way constitute the associated language called a pattern language.

Pure Grammars

Maurer et al. [34] introduced a different type of grammar called pure grammar. The study on pure grammars and pure languages [20, 34, 38] has been referred
by many researchers. In pure grammars, there is no distinction between non-terminals and terminals but the rewriting process is sequential as in the Chomskian grammars. This means that all the intermediate words in a derivation are necessarily in the language generated and consequently such languages differ considerably from languages generated by grammars where variables can be used to exclude words from the language.

**Grammar Systems**

In the last few decades, the theory of grammar systems has widely been investigated and presently it constitutes a well-developed formal language theory that presents several advantages over classical models. Theoretical studies in grammar systems have provided interesting results regarding generative capacity, descriptional complexity, decidability properties, etc. A variety of grammar systems has been considered in the literature and among these, parallel communicating grammar system has been taken up for studies by many researchers. Parallel communicating grammar systems (PC grammar systems, in short) were introduced by Păun et al. [40] as a grammatical model of parallelism in broad sense. A compact account of many of these details is provided by Dassow et al. [15] and Martin-Vide et al. [31]. A PC grammar system consists of several grammars working synchronously, each on its own sentential form and communicating by request. In a PC grammar system, each component has its own sentential form. Within each time unit (there is a clock common to all component grammars) each component applies a rule, rewriting its own sentential form. The key feature of a PC grammar system is its
“communication through queries” mechanism. Special (query) symbols are provided with each symbol pointing to a component of the system. When a component introduces (generates) the query symbol $Q_j$, the current sentential form of the component $j$ will be sent to the component $i$, replacing all the occurrences of $Q_j$. One component of the system is designated as master, and the language it generates is the language of the system. Several variants of the communication mechanism can be considered. They are determined by such things as shape of the communication graph, or the action a component has to perform after communication. Demetrescu and Păun [17] have examined in detail the power of PC grammar systems with right-linear components. Parallel communication has been subsequently investigated from different points of view [26, 29].

Parallel communicating grammar systems with communication by commands by Csuhaj-Varju et al. [13], represent the first model of networks of language processors where communication is performed through filters. A rewriting step in these systems is defined as follows. Each grammar generates its own string until it has no more applicable productions. Then the components communicate their strings to each other in the following manner. Every grammar tries to send a copy of its string to each of the other grammar, but only those strings are accepted at a component which passes through the filter associated with it.

A cooperating distributed grammar system (CD grammar system, in short) is sequential. Here all the component grammars have a common sentential form. Initially, there is a common axiom. At each moment, only one grammar is active - it rewrites the current sentential form. The matters such as, which
component grammar can become active at a given moment, and when an active grammar becomes inactive leaving the current sentential form to the other component grammars, are determined by the cooperation protocol. Examples of stop conditions (becoming inactive) are, the active component has to work exactly for $k$ steps, atleast $k$ steps, atmost $k$ steps, or the maximal number of steps (a step means an application of rewriting rule). Many other stop conditions are considered in the literature.

**Splicing Grammar System**

Head et al. [25], in an analysis of certain biochemical processes involving DNA under the influence of restriction enzymes and ligases, introduced a new operation on strings called splicing. Since then his basic idea has been formalized in terms of generative mechanisms for formal languages called splicing systems. Mateescu et al. [32] have considered simple splicing systems that make use of splicing rules that are as simple as possible. Dersanambika et al. [18] examined simple splicing grammar systems using the simple splicing rules of Mateescu et al. [32]. Various properties of the resulting simple splicing grammar systems are obtained by considering different component grammars.

**Learning Regular Sets**

Asking queries when something is to be learnt is one of the most natural things to do. In machine learning, learning by “asking queries” was first modeled and
investigated by Angluin [6]. Since then learning with queries has been intensively studied by many researchers. The types of queries used are membership, equivalence, subset, superset, disjointness and exhaustiveness queries. A learning algorithm (query learner) receives information about a target concept by asking queries which will be answered by an oracle (minimally adequate teacher). After asking atmost finitely many queries, the learner is required to make up its mind and to output its one and only hypothesis. If this hypothesis correctly describes the target concept, learning is said to be complete.

Language identification in the limit by Gold [22] through positive examples and the problem of identifying an unknown regular set from examples of its members and nonmembers is of great interest. It is assumed that the regular set is presented by an oracle which can answer membership queries about the set and can also test a conjecture and indicate whether it is equal to the unknown set and provide a counterexample if not [5]. A learning algorithm that correctly learns any regular set with the help of oracle in time polynomial is given in [5].

**Summary of the Work done in this Thesis**

Motivated by the extensive investigation on pattern grammars [16], pattern systems [37] and pure pattern grammars [2], the concepts of synchronized and non-synchronized pure pattern grammars have been taken up for research in this thesis and are investigated [50] and compared with existing grammars.

Motivated by the work done on parallel communicating grammar systems [26, 40]
and communications by command [13], parallel communicating synchronized pure
pattern grammar systems with filters [1, 46] is introduced.

Based on the work of Dersanambika et al. [18] on simple splicing grammar
system, simple splicing pattern and synchronized pure pattern grammar systems
with four splicing rules namely $<1, 3>$; $<1, 4>$; $<2, 3>$ and $<2, 4>$ are
developed [44, 49].

The study of Angluin [5] on problem of identifying an unknown regular set from
examples of its members and nonmembers and the study on inductive inference
[7, 19, 53, 54] are some focal areas of interest. With this interest, learning algorithms
for subclasses of synchronized pure pattern languages, parallel communicating
synchronized pure pattern grammar systems and simple splicing pattern grammar
systems are designed.

Csuhaj-Varju et al. [12], in the study of grammar systems have given a compact
account of cooperating distributed (CD) grammar system which paved way to the
work on cooperating distributed pattern grammar system in this thesis.

This thesis has been divided into five chapters. The findings of this work have
appeared in [1, 3, 44, 45, 46, 47, 48, 49, 50, 51].

**Synchronized Pure Patterns Grammars**

In the first chapter [50] of the thesis, a variant of pure pattern grammar has been
developed, which was originally introduced by Abisha et al. [2]. This provides
a natural link between pure grammars [34] and pattern grammars [16, 39]. The
pure pattern grammar has only one kind of symbol namely, terminal symbol or constant, as in pure grammars. The generation of words involves a process that is analogous to that in a pattern grammar. In other words, the pure pattern grammar has patterns which are strings of constants or terminal symbols. The constants are replaced initially by axioms over terminal symbols. The process is continued by replacing at any step, the symbols in a pattern with the current set of words derived, thereby yielding the associated language. Two modes of working of a pure pattern grammar, which are referred to as synchronized and non-synchronized modes, have been introduced. The resultant families of synchronized and non-synchronized pure pattern languages are compared with other families of languages such as pattern languages [16], pure languages [34], Chomskian languages [43] and L system languages [41]. Certain closure properties and descriptional complexity measures in line with [23] for synchronized pure pattern grammars are also obtained.

**Synchronized Pure Pattern Grammar and Parallel Communication**

PC grammar systems by Păun et al. [40] have been studied by many researchers. A compact account of many of these details is provided in [15]. Demitrescu et al. [17] have examined in detail the power of PC grammar systems with right linear components. In formal language theory, studies related to language generation use techniques that are based on notions in DNA computing. One such notion is the concept of filter. Parallel communicating grammar systems with communication by commands of Csuhaj-Varju et al. [13] represent the first model of networks of
language processors where communication is performed through filters.

In the second chapter of the thesis, a variant in parallel communicating (PC) grammar systems with filters is introduced [1, 46]. The variation here is that the components are synchronized pure pattern grammars and the master is regular. A rewriting step in this system is defined as follows. Each synchronized pure pattern grammar generates its own string, till the master component produces a query symbol \( Q_j \). Then the corresponding component sends a copy of its string to the master component, but only those strings are accepted at the master component which pass through the filter associated with it. The string which is filtered is catenated with the string produced by the master component. The resultant families of languages are compared with other families of languages such as pattern languages, pure languages, Chomskian languages and \( L \) system languages.

**Splicing Pattern and Synchronized Pure Pattern Grammar Systems**

In the third chapter of the thesis, four simple splicing rules namely \(< 1, 3 >\), \(< 1, 4 >\), \(< 2, 3 >\) and \(< 2, 4 >\) are used in simple splicing synchronized pure pattern grammar and pattern grammar systems [44, 49]. In these grammar systems the master component is regular or context-free and the other components are pattern or synchronized pure pattern grammars. Also a system with only pattern grammars as components has been discussed. The generative power of the system is compared with that of other existing models.
Learning of Languages

Angluin [5] introduced the notion of “minimally adequate teacher” (MAT) and the teacher (Oracle) answers membership and equivalence queries in order to construct a learning algorithm for regular sets. Angluin [6] has also introduced the notions of subset and superset queries. For a subset (superset) query, the input is a concept $C$ and the output is ‘yes’ if $C$ is a subset (superset) of the target concept $C^*$ and ‘no’ otherwise. If the answer is ‘no’, a counter example $x$ from $C - C^*(C^* - C)$ is also returned. Restricted subset queries and restricted superset queries, where no counter example is returned are also considered in [6]. Learning pattern languages using queries by Matsumoto et al. [33] has been an inspiration for learning subclasses of synchronized pure pattern languages.

In the fourth chapter of the thesis, three algorithms, to learn subclasses of synchronized pure pattern languages using the restricted subset queries and restricted superset queries, have been described [3]. In fact, the class considered for learning is the family of synchronized pure pattern languages generated by synchronized pure pattern grammars with a single pattern. The inclusion problem for this class is decidable, as the difference between the pattern languages with a single pattern over only variables and the synchronized pure pattern languages of this class lies mainly in the patterns. After learning subclasses of synchronized pure pattern languages, learning algorithm for a kind of parallel communicating grammar systems is formulated [45, 48]. In this grammar system, the master component is a regular grammar and the remaining components are synchronized pure pattern
grammars. One learning algorithm for simple splicing grammar systems with pattern grammars as components is also developed [51].

**Pattern Grammar and Cooperating Distributed Grammar Systems**

Motivated by pattern grammars of Dassow et al. [16] and cooperating distributed grammar systems by Csuhan-Varju et al. [12], a new grammar system, called cooperating distributed pattern grammar system (CDPGS) is introduced in the fifth chapter [47]. In this system all the components considered are pattern grammars. The resultant family of languages is compared with other families of languages and interesting results have been obtained. Also an algorithm is developed to learn a cooperating distributed pattern grammar system.