Variants in Synchronized Pure Pattern Languages

Sindhu J Kumaar, P. J. Abisha,

Abstract: A language generative device, known as Synchronized Pure Pattern grammar, is considered. This serves to link the notions of Pure grammar and Pattern grammar that have been introduced and investigated in the literature with different motivations. Three different variants of synchronized Pure Pattern grammars which was introduced earlier is considered and resultant families of languages are compared for their generative power with certain well-known families of languages
Index Terms:

1. Introduction

Formal language theory, which is one of the foundation areas of theoretical computer science, is replete with an abundance of grammars that have been introduced and investigated with different motivations. The pure grammars introduced in]7] is more in line with early work of Thue on words[8] in the sense of not dividing the alphabet into terminals and non-terminals unlike the well-known [12] Chomskian grammars. Also in contrast to the well-investigated L systems which involve rewriting in parallel, the rewriting process in a pure grammar is sequential as in the Chomskian grammars. A number of investigations on pure grammars in terms of theoretical properties and applications have been done in the literature.

On the other hand a different kind of language generative model, called pattern grammar, was introduced in [5]. This grammar involves an operation of replacing, by a special set of strings in parallel, all variables in a pattern string. The replacement is done in a uniform way in the sense of replacing all occurrences of the same variable in a pattern by the same string. This kind of grammar is motivated by the study of Angluin [1] on patterns that describe a set of strings.

Pattern grammars have been subsequently investigated from different points of view.

In this paper we consider a new generative device known as a synchronized pure pattern grammar, which was originally introduced in [14]. This provides a natural link between pure grammars [7] and pattern grammars [5] which had motivations from different directions. 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 synchronized pure pattern grammar has patterns which are the strings of constants of 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, there by yielding the associated language. We introduce three modes of working of a synchronized pure pattern grammar

2 Preliminaries

We recall some necessary definitions. For unexplained notions and notations, we refer to [13]. An alphabet Σ is a finite set of symbols. A word over Σ is a finite sequence of symbols of Σ. The set of all words over Σ is denoted by Σ^* which includes the empty word λ. We write $\Sigma^+ = \Sigma^* - \{\lambda\}$

Definition 1[9] A pure grammar is a triple G = (Σ, P, S) where Σ is a finite alphabet, S is a finite set of words over Σ and P is a finite set of ordered pairs (x, y) of words over Σ. The elements of P are referred to as productions, usually written as $x \to y$. If $x \in \Sigma^+$, in every production $x \to y$ of P, then G is called a pure context-free grammar (PCF)

In a pure grammar, a word w over Σ yields directly a word w′ over Σ according to G if there are words $w_1, w_2 \in \Sigma^*$ and a production $x \to y$ in P such that $w = w_1xw_2$ and $w' = w_1yw_2$. We then write $w \Rightarrow_G w'$ or briefly $w \Rightarrow w'$ (if G is understood). The reflexive, transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$. The language L(G) generated by G, called pure language, is defined as

L(G) = {w / s $\Rightarrow^*$ w, for some s in S}. The language generated by a PCF grammar is called PCF language.

Example 1

The PCF grammar G = ({a,b},{a → ab}, {a}) generates the PCF language L(G) consisting of all words of the form ab^n, n = 0, 1, 2, …

Though in both the Chomskian and pure grammars, the rewriting process is sequential, it is known [13] that these two families of languages are incomparable.

Definition 2 [5, 11] A pattern grammar is a 4 - tuple G = (Σ, X, A, P) where Σ is an alphabet whose elements are called constants, X is an alphabet whose elements are called variables, A ⊆ Σ* is a finite set of words, called axioms, P ⊆ (Σ ∪X)* is a finite set of words called patterns where each word contains at least one variable.
The rewriting in G is defined as follows: Initially, words are obtained by replacing in parallel and uniformly all the variables in a pattern in P by axioms of A with different occurrences of the same variable being replaced by the same word. The process is continued in a similar way by replacing variables by words from the current set of strings obtained. The language generated by G is L (G) = A ∪ P(A) ∪ P(P(A)) ∪…, where P(X) denotes the set of strings obtained from patterns in P by using strings of the set X in the manner described above.

Example 2 G = ({a, b} , {δ} , {ab} , {aδb}) is a pattern grammar generating the language L(G) consisting of all words of the form a^nb^n, n = 1, 2, …. In fact A = {ab} and initially the axiom word ab replaces δ in the pattern aδb to yield aabb. The process is repeated. Thus P(A) = {aabb}, P(P(A)) = {aaabbb } and so on.

It is known [5] that the pattern grammars generate a family of languages incomparable with Chomskian languages [213 and Lindenmayer languages.

3 Variants of Synchronized Pure Pattern Grammars

In this section, we consider the notion of a synchronizing pure pattern grammar (SPPG) and non-synchronizing pure pattern grammar (NSPPG). Synchronized Pure pattern grammars (SPPG) were originally introduced in [14] linking the studies of pure grammars [7] and pattern grammars [5]. Here we recall the definition synchronized pure pattern grammar (SPPG). Also we define non-synchronizing grammars (NSPPG).

Definition 3 A synchronized pure pattern grammar (SPPG) is a triple G = (Σ, A, P) where Σ is an alphabet, A ⊆ Σ* is a finite nonempty set of elements of Σ*, called axioms and P is a finite nonempty subset of Σ^+, called the set of patterns. For a set P and a set of words X ⊆ Σ*, let P(X) be the set of strings obtained by replacing all letters of every pattern by strings in X, uniformly and in parallel. Different occurrences of the same letter in a pattern are replaced by the same string.

Initially the symbols in a pattern are replaced by the axioms and subsequently the replacement process is continued with the set of words obtained at the current step. The language (SPPL) generated by G, denoted by L (G), is the smallest language L ⊆ Σ* for which we have P ⊆ L, A ⊆ L and P (L) ⊆ L. In fact L(G) = P ∪ A ∪ P(A) ∪ P(P(A)) ∪ …. We denote by SPPL itself the family of languages generated by SPPGs.

Non Synchronized Grammars

Definition 4 A non-synchronized pure pattern grammar (NSPPG) is a triple G = (Σ, A, P) where Σ is an alphabet A ⊆ Σ* is a finite nonempty set of elements of Σ* called axioms and P is a finite nonempty subset of Σ^+ called the set of patterns. The difference in the working of a NSPPG is that at the r^{th} step, each letter of the pattern is replaced by words from $\bigcup_{i=0}^{r-1} P^i(A)$ unlike in SPPG where at the r^{th} step each letter of the pattern is replaced by words $P^{r-1}(A)$.

In other words we start with the axiom set A and use the words of the axiom set in the replacement of symbols in a pattern in P to obtain P(A). We

then use words in A ∪ P(A) to obtain P(A ∪ P(A)) and the process is continued. Thus the language of the NSPPG G is L(G) = P ∪ A ∪ P(A) ∪ P(A ∪ P(A)) ∪ P(A ∪ P(A) ∪ P(A ∪ P(A))) ∪ …. We denote the family languages generated by NSPPG by NSPPL.

Remark 1: Note that in a SPPG and a NSPPG, the patterns themselves are in the language of the grammar. Also note that there is no difference in the components of a SPPG and a NSPPG. The difference lies only in the working.

Example 3 We illustrate with some SPPGs and NSPPGs with the corresponding languages generated.

i) G_1 = ({a}, {a}, {aa}). Here L (G_1) = $\{a^{2^n} / n = 0,1,2,...\}$. In fact initially the axiom a replaces both a's in the pattern aa to yield a^2 which is then used to replace again both a's in the pattern aa giving a^4 and the process continues. This grammar in non-synchronizing mode generates the same language.

ii) G_2 = ({a, b}, {λ, a, b}, {ab}). L(G_2) = Σ*. Any of the axioms λ, a, b can initially replace independently a as well as b in the pattern ab yielding λ, a, b, aa, ab, ba, bb. The resulting words can be used in a similar manner in the pattern ab and the process can be continued to yield the language consisting of all strings over a, b including the empty string.

iii) Consider G_3 = ({a, b}, {a}, {abb}). In synchronizing mode the language generated is L(G_3) = $\{a^{3^n} / n = 1,2,...\}$ ∪{abb}. In non-synchronizing mode in the first step, using a we obtain a^3 and in the next step we obtain a^3, a^5, a^7, a^9. Note that if we work in the synchronizing mode we can obtain only a^9. In the subsequent step in the non-synchronizing mode we obtain a^{11}, a^{13}, a^{15}, a^{17}, a^{19}, a^{21}, a^{23}, a^{25}, a^{27}. Thus the language generated in non-synchronizing mode is L(G_3) = {abb} ∪ {a^{2n-1}/ n = 1, 2, 3, …}.

Here we consider two more variants in the above formalism of defining SPPGs. In the first case, we do not include the set of patterns in the language. In the second, we allow rewriting of patterns using patterns, not only the axioms. The following discussion shows what happens in these cases and compare them with definition 3.

Case (i): A Synchronizing pure pattern grammar $SPPG_1$ is a triple G = (Σ, A, P) where Σ is an alphabet, A ⊆ Σ* is a finite nonempty set of elements of Σ*, called axioms and P is finite nonempty subset of Σ+, called the set of patterns. For a set of words X ⊆ Σ*, let P(X) be the set of strings obtained by replacing uniformly and in parallel, all the letters in every pattern of P, by strings in X. Different occurrences of the same letter in a pattern are replaced by the same string. Initially the symbols in a pattern are replaced by the axioms and subsequently the replacement process is continued with the words obtained at the current step. The language SPPL, generated by G, denoted by L(G), is the smallest language L ⊆ Σ* for which we have A ⊆ L and P(L) ⊆ L.

In fact L(G) = A ∪ P(A) ∪ P(P(A)) ∪ …

If $SPPG_1$ G is defined as above then the language $SPPL_1$ generated in synchronizing mode becomes subclass of a pattern language (PL). Reason is that as the pattern is not included in the language defined by a $SPPG_1$ G, any language generated in synchronizing mode can be generated by a pattern grammar, but the converse is not true. This is seen from the following examples.

Example 4:

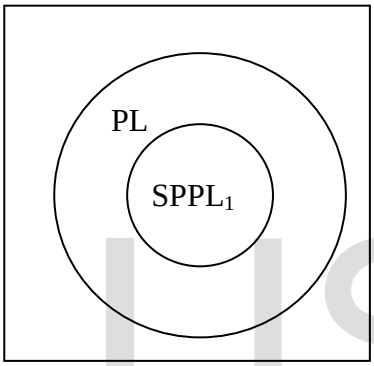
(i) Consider a synchronized pure pattern Grammar G_1 = ({a,b}, {a, b}, {ab}) then P(A) = {a^2, b^2, ab, ba}; P(P(A)) = {a^4, a^2b^2, a^3b, a^2ba, b^2a^2, b^4, b^2ab, b^3a, aba^2, ab^3, abab, ab^2a, ba^3, bab^2, ba^2b, baba} …

Same language can be generated by a Pattern Grammar G = ({a,b}, {a, b}, { δ_1 δ_2 }) then

P(A) = {a^2, b^2, ab, ba}; P(P(A)) = {a^4, a^2b^2, a^3b, a^2ba, b^2a^2, b^4, b^2ab, b^3a, aba^2, ab^3, abab, ab^2a, ba^3, bab^2, ba^2b, baba} …

If the pattern in $SPPG_1$ is {aa} then correspondingly in PG a pattern of the form $\{\delta\delta\}$ can be used. So if $SPPL_1$ is defined as $L(G) = A \cup P(A) \cup P(P(A)) \cup \ldots$ then $SPPL_1 \subseteq PL$ as such construction can be done for any PPG.

(ii) Consider the pattern language $L(G_2) = \{a^n b^n / n = 1, 2, \ldots\}$ generated by pattern grammar $G_2 = (\{a,b\}, \{\lambda\}, \{a\ \delta\ b\})$. By definition of a $SPPG_1$ we know the $L(G_2)$ cannot be generated by any $SPPG_1$.



Case (ii): If $SPPL_2$ is defined as $\mathbf{L(G) = P \cup A \cup P(A) \cup P(P(A)) \cup \ldots .. P(P) \cup P(P(P)) \cup \ldots}$ then in this case the pattern substitutions on pattern are included in the language. If the language $SPPL_2$ generated by $SPPG_2$ G is defined as above then $SPPL_2$ becomes incomparable with PL but not disjoint. This is seen from the following example.

Example 5:

(i) Consider a PPG G = ({a,b}, {a, b}, {ab}) then P(A) = {a^2, b^2, ab, ba}; P(P) = {abab}; P(P(A)) = {a^4, a^2b^2, a^3b, a^2ba, b^2a^2, b^4, b^2ab, b^3a, aba^2, ab^3, abab, ab^2a, ba^3, bab^2, ba^2b, baba}; P(P(P)) = {abababab} …

(ii) But in Pattern Grammar the string {ab} is included in the axiom set, let G = ({a,b}, {a, b, ab}, $\{\delta_1\ \delta_2\}$) then P(A) = {a^2, b^2, ab, ba, abab, a^2b, bab, aba, ab^2}; P(P(A)) = {a^4, a^2b^2, a^3b, a^2ba, b^2a^2, b^4, b^2ab, b^3a, aba^2, ab^3, abab, ab^2a, ba^3, bab^2, ba^2b, baba, abababab, $ababa^2$, $abab^3$, ababab, $abab^2a$, $ababa^2b$, ..} …

Here the strings abab, abababab which are in SPPL generated by pattern substitutions on P are in PL, but strings like a^2b, bab, aba are not in SPPL. Thus if SPPL is defined as $\mathbf{L(G) = P \cup A \cup P(A) \cup P(P) \cup P(P(A)) \cup P(P(P))}$ … certain strings generated by PG cannot be generated by SPPG hence SPPL becomes incomparable but not disjoint with PL. Any way, it is unusual to substitute patterns into patterns, as only axioms should be the start point of rewriting process.

4. Conclusion

We have considered here three modes of derivation of synchronized pure pattern grammar. We have shown three variants in SPPL of which one is a subclass of pattern language PL and the other is incomparable and disjoint with pattern language PL, but the model which we have considered is novel in the sense it is not a subclass of pattern language and does not involve pattern substitutions on pattern set P.

References

[1] Angluin, D. (1980). Finding patterns common to a set of strings, *Journal of Computer and System Sciences*. **21**, 46 – 62.

[2] Berstel, J. (1995). Axel Thue's Papers on Repetitions in Words: a Translation, *Publications du LaCIM, Dpartement de mathmatiques et d'informatique*, **20**.Universit du Qubec Montral.

[3] Castiglione, G., Restivo, A. and Salemi, S. (2004). Patterns in words and languages, *Discrete Appl. Math*, **144**, 237 - 246.

[4] Crepinsek, M., Kosar, T., Mernik, M., Cervelie, J., Forax, R., Rousse, G. (2010). On Automata and Language Based Grammars Metrics, *Computer Science and Information Systems*, **7(2),** 309 – 329.

[5] Dassow, J, Paun, G. and Salomaa, A. (1993). Grammars based on patterns, *International Journal of Foundations of Computer Science* **4**, 1-14.

[6] Gabrielian, A. (1981). Pure grammars and pure languages, *Int. J. Comput. Math*. **9**, 3 - 16.

[7] Maurer, H. A., Salomaa, A. and Wood, D. (1980). Pure Grammars, *Journal of information and Control*, **44**, 47 – 72.

[8] Mitrana, V. (1996). Iterated pattern languages, *J. Autom. Lang. Comb.*, **1,** 305 - 311.

[9] Mitrana, V. (1999). Patterns and languages: An Overview, *Grammars*, **2**, 149 – 173.

[10] Mitrana, V., Paun, Gh., Rozenberg, G. and Saloma, A. (1996). Pattern systems, *Journal of Theoretical Computer Science*, **154**, 183 - 201.

[11] Paun, Gh., Rozenberg, G. and A. Salomaa, (1996). Pattern Grammars, *Journal of Automata, Languages and Combinatorics*, **1**, 219 – 235.

{12} Rozenberg G. and Saloma, A. (1997). Hand book on Formal Languages, Volume 1 – 3, Springer Verlag, The Mathematical Theory of L – Systems, Academic Press, New York.

[13] Salomaa, A. (1973). Formal Languages, *Academic Press*, New York.

[14] Sindhu J Kumaar, Abisha, P. J., Thomas, D. G., Nor Haniza Sarmin, Subramaniam, K. G. (2013). Languages defined by Pure Pattern Grammars, accepted for publication in International Journal of Applied Mathematics and Computational Inteligence.

[15] Subramanian, K. G., Ali, R. M., Geethalakshmi, M., Nagar, A. K. (2009). Pure 2D picture grammars and languages, *Discret Appl. Math.*, **157,** 3401 – 3411.