Journal of Mathematics and Informatics
Vol. 7, 2017, 1-6
ISSN: 2349-0632 (P), 2349-0640 (online)
Published 24 February 2017
www.researchmathsci.org
DOI: http://dx.doi.org/10.22457/jmi.v7a1

Journal of
Mathematics and
Informatics

Alphabetic Flat Splicing Pure Context-free Grammar Systems

G. Samdanielthompson, N.Gnanamalar David* and *K. G. Subramanian

Department of Mathematics, Madras Christian College
Chennai – 600 059, India
E-mail:{samdanielthompson, ngdmcc, kgsmani1948 }@gmail.com

Received 2 February2017; accepted 23 February2017

Abstract. In the context of DNA computing, an operation on words, called splicing, was introduced by Head (1987), in his study on recombinant behavior of DNA molecules. Recently, a special kind of splicing, called flat splicing on strings, was considered and several properties were derived. On the other hand, in the study of modelling of distributed complex systems based on language theory, grammar systems were proposed. Here we introduce a grammar system, called alphabetic flat splicing pure context-free grammar system ($AFSPCFGS$), as a new model of language generation, based on the operation of alphabetic flat splicing on words and pure context-free rules. We derive certain comparison results that bring out the generative power of $AFSPCFGS$ and as an application construct a $AFSPCFGS$ generating "floor-design" pictures.

Keywords: Flat splicing, formal language, grammar system, pure grammar

AMS Mathematics Subject Classification (2010): 68Q42

1. Introduction

The theory of formal grammars and languages is considered to be a fundamental branch of theoretical computer science. The notion of grammar system [3] consisting of several grammars that co-operate according to some well-defined protocol is an established area of study, motivated by the idea of modeling distributed complex systems. The general idea of a grammar system is to have components, each with a certain type of grammar rules, and there are different types of communication between components, allowing the generation of different classes of languages.

On the other hand splicing on words that are strings of symbols is a bio-inspired operation, introduced by Head [4], in his study of modelling the recombinant behaviour of DNA molecules under restriction enzymes and ligase in the context of DNA computing. This operation was utilized in developing theoretical models of computation in the framework of formal language theory [8], which is considered to be the backbone of theoretical computer scienceand has relevance in many fields such as combinatorics on words [5,7]. Based on this splicing operation, significant theoretical results on computational universality, have been established. Recently, flat splicing on words has

been introduced by Berstelet al. [1], as a special kind of splicing. This operation acts on a pair (u, v) of words over an alphabet Σ , "cutting" u and "inserting" v into it, as directed by a flat splicing rule of the form $(\alpha|\gamma\ -\delta|\beta)$so that $u\ =\ x\ \alpha\ \beta\ y$ and $v\ =\gamma\ z\ \delta$, for words $x, y, z, \alpha, \beta, \gamma, \delta$ over $\Sigma$and the resulting word is $w\ =\ x\ \alpha\ \gamma\ z\ \delta\ \beta\ y$. If $\alpha, \beta, \gamma, \delta$ are either letters of the alphabet or the empty word, then the rule is called alphabetic. Several properties associated with flat splicing in the context of grammars and languages, have been studied in [1].

In the theory of formal grammars and languages, classes of grammars specified by different control mechanisms have been introduced to regulate rewriting and thereby increase the language generative capacity of grammars. Pure context-free grammars [6] which make use of only one kind of symbols, called terminal symbols, have been investigated in formal string language theory for their generating power and other properties. It is known that there are regular languages that cannot be generated by any pure context-free grammars. Recently, using the flat splicing operation on words for the communication between components and context-free or regular rules in the components, a variant called flat splicing grammar system has been introduced [2] and studied. Here with pure context-free rules in the components and alphabetic flat splicing rules for ``communication" between components, we introduce an alphabetic flat splicing pure context-free grammar system ($AFSPCFGS$) as a new model of language generation. The language class of $AFSPCFGS$ is compared with certain other language classes and as an application we construct a $AFSPCFGS$ for describing certain ``floor designs".

2. Preliminaries

We refer to [8], for concepts and results related to formal grammars and languages. In this section, we recall some basic notions and results.

An alphabet $\Sigma$is a finite set of symbols. A word w over Σ is a finite sequence of symbols. We denote by Σ^*, the set of all words over Σ, including the empty word λ with no symbols and write $\Sigma^+\ =\Sigma^*\ -\ \{\lambda\}$. The length $|w|$ of a word w is the number of symbols in w counting repetitions of symbols in w. Clearly, $|\lambda|\ =\ 0$.

We now recall the notion of flat splicing on words [1]. The idea flat splicing is that a word with a specified ``prefix" and a specified ``suffix" is inserted into another word in a pre-determined position. Expressed in formal terms, a flat splicing rule r is of the form $(\alpha\ |\gamma\ -\delta\ |\beta\ )$, where $\alpha, \beta, \gamma, \delta$ are words over an alphabet Σ. For two words $=\ x\alpha\ \beta\ y;\ v\ =\gamma\ z\ \delta$, an application of the flat splicing rule $r\ =\ (\alpha\ |\gamma\ -\delta|\ \beta)$ to the pair (u, v) yields the word $w\ =\ x\alpha\ \gamma\ z\ \delta\ \beta\ y$ and we write $(u, v)\vdash_r\ w$. A flat splicing rule $r\ =\ (\alpha\ |\gamma\ -\delta\ |\beta)$, where $\alpha, \beta, \gamma, \delta$ are letters in Σor the empty word, is called alphabetic. A flat splicing system (FSS) [1] is a triple $S\ =\ (\Sigma, I, R)$, where Σ is an alphabet; I, called the initial set, is a set of words over Σ, and R is a finite set of flat splicing rules [1]. The $FSS\ S$ is respectively called finite, regular or context-free according as I is a finite set, regular set or a context-free language. The language L generated by S is the smallest language containing I and such that for any two words $u, v\in\ L$, the word w is also in L, if $(u, v)\vdash_r\ w$. When all the flat splicing rules are alphabetic, the FSS is called an alphabetic flat splicing system ($AFSS$). The families of languages generated by FSS and $AFSS$ are respectively denoted by $L(FSS, X)$ and

$L(AFSS, X)$ for $X = FIN, REG$ or CF according as the initial set is finite, regular or context-free.

Definition 2.1. A pure context-free grammar [6]$(PCF)$is $G = (\Sigma, P, A)$, where Σ is a finite alphabet, A is a finite set of axiom words and P is a finite set of ordered pairs (a, y), referred to as productions and usually written$a \rightarrow y$,where $a \in \Sigma$ and $y \in \Sigma^*$.

Derivations are done as in a context-free grammar except that, unlike a Chomsky context-free grammar, there is only one kind of symbol namely the terminal symbol. The language generated consists of all words generated from each of the axiom words. The family of languages generated by PCF grammars is denoted by $L(PCF)$.

Example 2.1. The pure context-free grammar$G = (\Sigma = \{x, y, d, P = \{d \rightarrow xdy\}, A = \{xdy\})$ generates the language $\{x^n dy^n | n \geq 1\}$. In fact a derivation in this grammar starts from the axiom word xdy and application of the rule $d \rightarrow xdy$for $(n-1)$, $(n \geq 1)$, times yields the word $x^n dy^n$. The language generated is $L(G) = \{x^n dy^n \mid n \geq 1\}$.

Remark 2.1. It has been shown [6] that the language $L_1 = \{a^n b^n \mid n \geq 1\}$ cannot be generated by any pure context-free grammar.

3. Alphabetic flat splicing pure context-free grammar systems

Flat splicing grammar systems with context-free or regular rules in the components have been considered in [2]. In fact essentially, alphabetic flat splicing rules are considered in [2], especially in the proofs of the results, although this is not explicitly mentioned. We refer to these as alphabetic flat splicing context-free or regular grammar systems. When the number of components is $n,\ n \geq 1$, we denote the generated corresponding families of languages respectively by $L_n(AFSCFGS)$ and $L_n(AFSRGS)$. We now introduce a variant called an alphabetic flat splicing pure context-free grammar system which has pure context-free rules in the components. Rewriting is done in parallel in the components but two different components "communicate" by alphabetic flat splicing rules. We now formally define this grammar system.

Definition 3.1. An alphabetic flat splicing pure context-free grammar system $(AFSPCFGS)$ is a construct $\Gamma = (\Sigma, (P_1, S_1), \cdots, (P_n, S_n), F)$ whereΣ is a finite set of symbols; P_i, for each $i = 1,2, \cdots, n$, is a finite set of context-free rules of the form $a \rightarrow \alpha, a \in \Sigma, \alpha \in \Sigma^*$; S_i, for each $i = 1,2, \cdots, n$, is a finite set of axioms over Σ;F is a finite set of alphabetic flat splicing rules.

Without loss of generality, we take the language generated by the first component as the $language$ of Γ. The family of languages generated by the alphabetic flat splicing pure context free grammar systems with at most n components is denoted by $L_n(AFSPCFGS)$.

We give an example of $AFSPCFGS$ of degree 2.

Example 3.1. Consider the alphabetic flat splicing pure context-free grammar system $AFSPCFGS$ of degree 2, given by

$$\Gamma = (\{a,b,c,d,e\},(\{c \rightarrow ac\},\{ace\}),(\{d \rightarrow bd\},\{bd\}),\{(c|b-d|e)\}).$$

In the first component, starting with the axiom word ace, application of the rule $c \rightarrow ac, (n-1)$ times generates the word $a^n ce$, $n \geq 1$ while at the same time, in the second component from the axiom word bd, application of the rule $d \rightarrow bd$, $(n-1)$ times generates the word $b^n d, n \geq 1$. If the alphabetic flat splicing rule $(c|b-d|e)$ which is applicable, is applied on the pair $(a^n ce, b^n d)$, then the word $a^n cb^n de$ is derived in the first component. While alphabetic flat splicing rule is no longer applicable, application of the rule $c \rightarrow ac$ can also be continued on the word $a^n cb^n de$ to yield words of the form $a^{\{n+m\}} cb^n de, n \geq 1, m \geq 0$. The language generated is

$$L(\Gamma) = \{ a^n ce \mid n \geq 1 \} \cup \setminus\{a^{\{n+m\}} cb^n de \mid n \geq 1, m \geq 0 \}.$$

Theorem 3.1. $L(PCF) = L_1(AFSPCFGS) \subset L_2(AFSPCFGS)$.

Proof: The inclusion $L_1(AFSPCFGS) \subseteq L_2(AFSPCFGS)$ follows from the definition of $AFSPCFGS$. In order to prove the proper inclusion, consider the language $L_2 = \{ xb^n y, zb^n y \mid n \geq 1 \} \cup \{ za^n b^n y \mid n \geq 2 \}$. The language L_2 is generated by the $AFSPCFGS$ of degree 2, given by

$$\Gamma_2 = (\{x,y,z,a,b\},(\{x \rightarrow xb, x \rightarrow zb\},\{xby\}),(\{a \rightarrow aa\},\{a\}),\{(z|a-a|b)\}).$$

In the first component, starting with the axiom word xby, application of the rule $x \rightarrow xb, (n-1)$ times followed by the rule $x \rightarrow zb$ generates the word $zb^n y, n \geq 1$ while at the same time, in the second component from the axiom word a, application of the rule $a \rightarrow aa$,n times generates the word $a^n, n \geq 1$. If the alphabetic flat splicing rule $(z|a-a|b)$ which is applicable, is used on the pair$(zb^n y, a^n)$, then the word $za^n b^n y$ is derived in the first component after which the alphabetic flat splicing rule is no longer applicable. On the other hand, the language L_2 cannot be generated by any pure context-free grammar, by an argument analogous to proving that the language $\{a^n b^n \mid n \geq 1\}$cannot be generated by any pure context-free grammar, as mentioned in Remark 2.1.

Corollary 3.1. There exists a context-free language which cannot be generated by any $AFSPCFGS$ with one component but can be generated by a$AFSPCFGS$ with two components.

Proof: This is a consequence of the context-free language L_2 in the proof of Theorem 3.1 wherein it is shown that L_2 is not in $L_1(AFSPCFGS)$ while it is generated by the $AFSPCFGS\Gamma_2$ with two components.

Theorem 3.2. There exists a context-sensitive language which is not context-free that can be generated by a $AFSPCFGS$ with two components.

Proof: Consider the language

$L_3 = \{ a^n cb^n, a^n eb^n \mid n \geq 1 \} \cup \{ a^n ed^{n+1} b^n \mid n \geq 1 \}$. This context-sensitive language is not a context- free language and is generated by the $AFSPCFGS$

$\Gamma_3 = (\{a,b,c,d,e\},(\{c \rightarrow acb, c \rightarrow e\},\{acb\}),(\{d \rightarrow dd\},\{d\}),\{(e|d-\lambda|b)\})$.

While in the first component, a word of the form $a^n eb^n$ is generated, for the same n, a

word d^{n+1} is generated in the second component and at this point the rule $(e|d-\lambda|b)$ is applicable. Hence flat splicing yields the word $a^n e d^{n+1} b^n$. Note that words $a^n c b^n$ are also generated in the first component but these words cannot be flat spliced with d^{n+1}.

4. An application

Array generating two-dimensional grammar models have been utilized [9] to generate certain types of picture patterns, such as "floor designs" (or also called "kolam patterns") treating the picture pattern as an array over terminal symbols, generating the array with an array grammar and then substituting for each symbol some relevant primitive pattern, yielding the pattern. Here we can utilize this technique in order to generate certain linear "kolam patterns" using the $AFSPCFGS$. The linear "kolam patterns" as shown in Fig.4.1 (B) (Right)constructed from the primitives shown in Fig. 4.1(A) (left) can be generated by substituting in the words $za^n b^n y$ (generated by the $AFSPCFGS$ in the proof of Theorem 3.1) the primitives for the respective symbols in Fig. 4.1. (A).

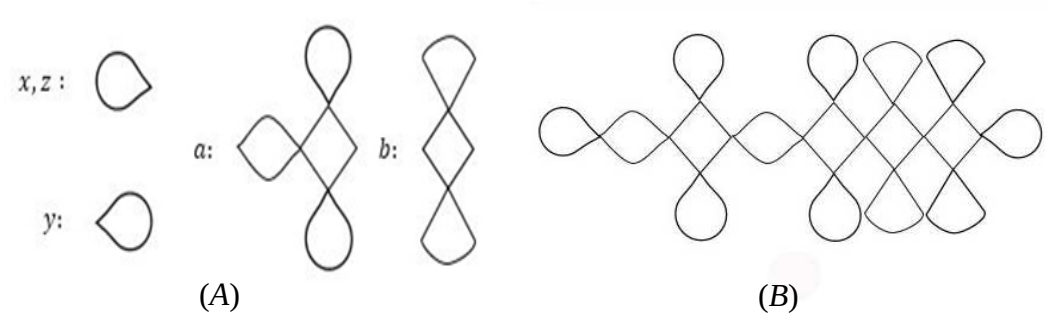

Figure 4.1: (A) Primitives (B) Kolam Pattern

Acknowledgements. The authors are grateful to the reviewers for their useful comments. The first author G. Samdaniel thompson acknowledges with gratitude the award (No.: F1-17.1/2016-17/MANF-2015-17-TAM-51257/ (SA-III/Website)) of Maulana Azad National Fellowship for minority students by UGC, India which has enabled him to carry out his research for PhD in the Department of Mathematics, Madras Christian College. The third author K.G. Subramanian is grateful to UGC, India, for the award of Emeritus Fellowship (No.F.6-6/2016-17/EMERITUS-2015-17-GEN-5933 / (SA-II)) to him to execute his work in the Department of Mathematics, Madras Christian College.

REFERENCES

1. J.Berstel, L.Boasson and I.Fagnot, Splicing systems and the Chomsky hierarchy. *Theoretical Computer Science*, 436 (2012) 2 -22.
2. R.Ceterchi and K.G.Subramanian, Grammar systems with context-free rewriting and flat splicing. In: Gheorghe et al. (eds.), M*ultidisciplinary Creativity*, Spandugino, (2015) 221-227.

3. E.Csuhaj-Varju, J.Dassow and J.Kelemen, Gh. Paun, Grammar systems: Agramatical approach to distribution and cooperation, Gordon and Breach Science Publishers, (1994).
4. T.Head, Gh. Paun, D.Pixton, Language theory and molecular genetics: generative mechanis suggested by DNA recombination, In: G. Rozenberg, A.Salomaa (Eds.), Handbook of Formal Languages Springer, Berlin, 2 (1997) 295-358.
5. M. Lothaire, *Combinatorics on Words*, Cambridge University Press, 1997.
6. H.A.Marurer, A.Salomaa and D. Wood, Pure Grammars, *Information Control*, 44 (1980) 47-72.
7. C.A.Priya Darshini, V. R. Dare, I. Venkat and K.G. Subramanian, Factors of words under an involution, *Journal of Mathematics and Informatics*, 1 (2013-14) 52-59.
8. G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages Springer, Berlin, Vol. 1-3, 1997.
9. G.Siromoney, R.Siromoney and K.Krithivasan, Array Grammars and Kolam. *Computer Graphics and Image Processing*, 3(1) (1974) 63-82.