Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A note on the class of languages generated by F-systems over regular languages

Jorge C. Lucero [a,*], Sławek Staworko [b]

[a] Dept. Computer Science, University of Brasília, Brasília, DF 70910-900, Brazil
[b] Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 - CRIStAL, F-59000 Lille, France

ARTICLE INFO

Article history:
Received 23 October 2020
Received in revised form 24 April 2022
Accepted 6 May 2022
Available online 11 May 2022
Communicated by Leah Epstein

Keywords:
F-system
Folding
Regular language
Linear language
Formal languages

ABSTRACT

An F-system is a computational model that performs a folding operation on words of a given language, following directions coded on words of another given language. This paper considers the case in which both given languages are regular, and it shows that the class of languages generated by such F-systems is a proper subset of the class of linear context-free languages.

1. Introduction

Geometric folding processes are ubiquitous in nature and technology, from the shaping of protein molecules [1] and the folding of leaves and insect wings [2], to self-assembling robots [3] and foldable space telescopes [4]. In current days, it is usual to designate such processes under the general term of "origami", in reference to the Japanese traditional art of creating figures by folding a sheet of paper [5].

From the perspective of the theory of formal languages, origami has been modeled by a word folding operation, which reorders symbols of a given word according to directions coded in another one [6]. Using the folding operation, a folding system (F-system) of the form $\Phi = (L_1, L_2)$ may be defined, where L_1 (the core language) is the language that contains the words to be folded, and L_2 (the folding procedure language) is the language that contains words with the folding directions. Although this model is restricted to one dimensional folding and does not capture actual origami (i.e., on a bidimensional sheet), it may still be applied to characterize folding processes in molecular or DNA computing and related areas [7–9].

The computing power of F-systems has been investigated by comparison with standard language classes from the Chomsky hierarchy (i.e., regular, context-free, context-sensitive, recursive and recursively enumerable languages). More recently [10], necessary conditions for a language to belong to classes generated when the core and the folding procedure languages are regular or context-free have been proposed in the form of pumping lemmas, similar to the well known pumping lemmas for regular and context-free languages.

The present paper considers the case in which both the core and the folding procedure languages are regular. It has been demonstrated that the class of languages generated by such F-systems surpasses and strictly contains the regular languages [6]. Here, it will be shown that the F-system class is a proper subset of the class of the linear context-free languages.

* Corresponding author.
E-mail addresses: lucero@unb.br (J.C. Lucero), slawomir.staworko@univ-lille.fr (S. Staworko).

https://doi.org/10.1016/j.ipl.2022.106283
0020-0190/© 2022 Elsevier B.V. All rights reserved.

2. Definitions

Let us first review the definitions of folding operations and systems [10].

Definition 1. Let Σ be an alphabet, $\Gamma = \{\mathtt{u}, \mathtt{d}\}$, and $f : \Sigma^* \times \Sigma \times \Gamma \to \Sigma^*$ a function such that

$$f(x, a, b) = \begin{cases} ax & \text{if } b = \mathtt{u}, \\ xa & \text{if } b = \mathtt{d}. \end{cases}$$

Then, the folding function $h : \Sigma^* \times \Gamma^* \to \Sigma^*$ is a partial function defined by

$$h(w, v) = \begin{cases} \varepsilon & \text{if } |w| = |v| = 0, \\ f(h(w', v'), a, b) & \text{if } |w| = |v| > 0, \\ & \text{with } w = w'a,\ v = v'b, \\ \text{undefined} & \text{if } |w| \neq |v|. \end{cases} \qquad \square$$

The computation of $h(w, v)$ may be regarded as a folding operation that rearranges the symbols of w. Words over Γ describe how each folding must be performed, where symbol $\mathtt{u}$ represents a "folding up" action and symbol $\mathtt{d}$ represents a "folding down" action (see [10] for an illustration of the folding mechanism).

Definition 2. A folding system (F-system) is a pair $\Phi = (L_1, L_2)$, where $L_1 \subseteq \Sigma^*$ is the core language, and $L_2 \subseteq \Gamma^*$ is the folding procedure language. The language of Φ is

$$L(\Phi) = \{h(w, v) |\, w \in L_1, v \in L_2, |w| = |v|\}. \qquad \square$$

Definition 3. The class of all languages generated by F-systems with core languages of a class $\mathcal{C}$ and folding procedure languages of a class $\mathcal{H}$ is

$$\mathcal{F}(\mathcal{C}, \mathcal{H}) = \{L(\Phi) |\, \Phi = (L_1, L_2), L_1 \in \mathcal{C}, L_2 \in \mathcal{H}\}. \qquad \square$$

We recall basic concepts of context-free and regular languages [11,12]. A *context-free grammar* is a tuple $G = (V, \Sigma, R, S)$, where V is the set of nonterminal symbols, Σ is the set of terminals, $R \subseteq V \times (V \cup \Sigma)*$ is the set of production rules, and $S \in V$ is the start symbol. G is *linear* if every production rule is of the form $A \to uBv$ or $A \to u$, where $u, v \in \Sigma^*$ and $A, B \in V$. G is right-linear if every production rule is of the form $A \to uB$ or $A \to u$, where $u \in \Sigma \cup \{\varepsilon\}$ and $A \in V$. The class of linear languages LIN consists of languages generated by linear grammars. The class of regular languages REG consists of languages generated by right-linear grammars.

3. Folding over regular languages

We consider languages of the class $\mathcal{F}(\mathsf{REG}, \mathsf{REG})$. First, we show that $\mathcal{F}(\mathsf{REG}, \mathsf{REG}) \subseteq \mathsf{LIN}$, where LIN is the class of linear languages.

Theorem 1. *The class of languages generated by F-systems with regular core and procedure languages is a subset of the class of linear languages.*

Proof. Consider the F-system $\Phi = (L_1, L_2)$ with $L_1, L_2 \in \mathsf{REG}$. Let $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Gamma, R_2, S_2)$ be right-linear grammars for reverse languages $L_1^{\mathcal{R}}$ and $L_2^{\mathcal{R}}$, respectively. Then, a linear grammar $G = (V, \Sigma, R, S)$ for $L(\Phi)$ may be obtained by letting:

1. $V = V_1 \times V_2$,
2. $R = R_{\mathtt{u}} \cup R_{\mathtt{d}} \cup R_{\varepsilon}$, where

$$R_{\mathtt{u}} = \{(A, B) \to a(C, D) \mid A \to aC \in R_1, \quad B \to \mathtt{u}D \in R_2\},$$
$$R_{\mathtt{d}} = \{(A, B) \to (C, D)a \mid A \to aC \in R_1, \quad B \to \mathtt{d}D \in R_2\},$$
$$R_{\varepsilon} = \{(A, B) \to \varepsilon \mid A \to \varepsilon \in R_1, \quad B \to \varepsilon \in R_2\},$$

3. $S = (S_1, S_2)$.

Now, for any nonterminal A of a grammar G, let G^A denote the version of G with A as the start symbol. With a straightforward inductive argument we prove the following claim.

Claim 1.1. For any $A_1 \in V_1$ and $A_2 \in V_2$, $L(L(G_1^{A_1})^{\mathcal{R}}, L(G_2^{A_2})^{\mathcal{R}}) = L(G^{(A_1,A_2)})$.

Naturally, the above claim proves that $L(\Phi) = L(G)$. $\square$

Example 1. Let $\Phi = (L_1, L_2)$ with $L_1 = (abc)^*$ and $L_2^* = (\mathtt{udd})^*$, and take the following right-linear grammars G_1 and G_2 defining $L_1^{\mathcal{R}} = (cba)^*$ and $L_2^{\mathcal{R}} = (\mathtt{ddu})^*$, respectively.

$$G_1 : S_0 \to \epsilon \mid cS_1 \qquad S_1 \to bS_2 \qquad S_2 \to aS_0$$
$$G_2 : T_0 \to \epsilon \mid \mathtt{d}T_1 \qquad T_1 \to \mathtt{d}T_2 \qquad T_2 \to \mathtt{u}T_0$$

The construction in the proof above yields the following linear grammar (nonproductive rules omitted).

$$G : (S_0, T_0) \to \epsilon \mid (S_1, T_1)c \qquad (S_1, T_1) \to (S_2, T_2)b$$
$$(S_2, T_2) \to a(S_0, T_0)$$

Clearly, $L(G) = \{a^n(bc)^n \mid n \geq 0\} = L(\Phi)$. $\square$

Now, we show that $\mathcal{F}(\mathsf{REG}, \mathsf{REG}) \neq \mathsf{LIN}$. The proof relies on an *interchange* property of languages generated by folding: if $w_1, w_2 \in L(\Phi)$, $|w_1| = |w_2|$, $w_1 = h(v_1, u_1)$, and $w_2 = h(v_2, u_2)$, then $h(v_1, u_2)$ also belongs to $L(\Phi)$. We construct a linear language that does not have this property.

Theorem 2. *The class of languages generated by F-systems with regular core and procedure languages is not equal to the class of linear languages.*

Proof. We present a linear language L that cannot be generated by any folding system with regular core and regular procedure languages. The language L over the alphabet

$\Sigma = \{a, b, c, d, e, f, \#\}$ is defined with the following linear grammar:

$$G : S \to S_1 \mid S_2, \qquad S_1 \to aS_1bc \mid a\#bc,$$
$$S_2 \to deS_2f \mid de\#f.$$

Suppose now that there is an F-system $\Phi = (L_1, L_2)$, such that $L = L(\Phi)$, and let N_1 and N_2 be the numbers of non-terminals of the right-linear grammars that define L_1 and L_2, respectively. We point out that L has only words of length $3i + 1$ for $i \geq 1$, and without loss of generality, we assume that both L_1 and L_2 have words of length $3i + 1$ only. Otherwise, we can take their intersections with the respective regular languages of words of length $3i + 1$.

Since every word in L has exactly one occurrence of #, so does every word in L_1. Moreover, with a pumping argument we show that # is in the beginning of every word in L_1. More precisely, we let $N = N_1N_2$ and make the following claim.

Claim 2.1. For every word $w \in L_1$, the symbol # is present in the first N symbols of w.

Next, let $n = 2N$ and take the words $w \in L_1$ and $v \in L_2$ such that $h(w, v) = a^n\#(bc)^n$. Note that $|w| = |v| = 6N + 1$. Let $w = w_1\#w_2$ and observe that since $|w_1| < N$, w_2 contains more than $3N$ symbols in $\{b, c\}$. Because those symbols follow #, they must be folded down, and therefore v must also contain at least $3N + 1$ occurrences of $\mathtt{d}$.

Now, take the words $w' \in L_1$ and $v' \in L_2$ such that $h(w', v') = (de)^n\#f^n$, and consider folding w' according to v (w' and v have the same length). Because w' contains only symbols in $\{d, e, f, \#\}$ the result $h(w', v)$ must also be equal to $(de)^n\#f^n$ (L demands it). However, we observe that $w' = w'_1\#w'_2$ and $|w'_1\#| \leq N$, and therefore, at least $2N+1$ symbols of w'_2 are folded down by v. Consequently, the result $h(w', v')$ has more than n symbols following #, which contradicts $h(w', v) = (de)^n\#f^n$. □

4. Conclusion

From Theorems 1 and 2, we conclude that $\mathcal{F}(\mathsf{REG}, \mathsf{REG}) \subset \mathsf{LIN}$. It is also known that $\mathsf{REG} \subset \mathcal{F}(\mathsf{REG}, \mathsf{REG})$ [6], which places $\mathcal{F}(\mathsf{REG}, \mathsf{REG})$ as an intermediate class between the regular and linear languages. Interestingly, Theorem 2 also shows that $\mathcal{F}(\mathsf{REG}, \mathsf{REG})$ is not closed under union: the linear language L used in the proof is the union of $L((abc)^*, (\mathtt{udd})^*)$ and $L((edf)^*, (\mathtt{uud})^*)$. Tackling the questions of closure under intersection and complement would require dedicated tools and we leave it for future work.

A previous work [10] introduced a weak pumping lemma stating conditions for a language to belong to $\mathcal{F}(\mathsf{REG}, \mathsf{REG})$. However, the present result implies that the class must also satisfy the pumping lemma for linear languages [13,14], which has stronger conditions than the previous lemma. The relation of the class with the linear languages also implies that it has efficient recognition algorithms of $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space complexities [15], which may be relevant for applications in natural computing.

It is also interesting to note that F-systems may be expressed in terms of families of permutations as defined in [16]. Since the even-linear languages [17], generated by linear grammars with rules $S \to uS'v$ such that $|u| = |v|$, may be obtained from permutations on regular languages [16, Example 9], then this class is contained within $\mathcal{F}(\mathsf{REG}, \mathsf{REG})$.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We are grateful to the anonymous reviewer who pointed out the relation of $\mathcal{F}(\mathsf{REG}, \mathsf{REG})$ with the class of even-linear languages. Jorge C. Lucero was supported by a grant from the Deans of Research and Innovation and of Graduate Studies of the University of Brasília.

Appendix A. Proof of Claim in Theorem 1

Proof. First, we show that any word $s \in L(L(G_1^{A_1})^{\mathcal{R}}, L(G_2^{A_2})^{\mathcal{R}})$ is also in $L(G^{(A_1,A_2)})$. If $s \in L(L(G_1^{A_1})^{\mathcal{R}}, L(G_2^{A_2})^{\mathcal{R}})$, then there are words $w \in L(G_1^{A_1})^{\mathcal{R}}$ and $v \in L(G_2^{A_2})^{\mathcal{R}}$ such that $|s| = |w| = |v|$ and $s = h(w, v)$, where h is the folding function defined in Definition 1. Using induction on the length of s:

1. If $|s| = 0$, then $s = w = v = \varepsilon$, and G_1 and G_2 have rules $A_1 \to \varepsilon$ and $A_2 \to \varepsilon$, respectively. Therefore, G has the rule $(A_1, A_2) \to \varepsilon$, and $\varepsilon \in L(G^{(A_1,A_2)})$.
2. If $|s| > 0$, then let $w = w'a$, $v = v'b$, where $a \in \Sigma$ and $b \in \Gamma$. Since $w^{\mathcal{R}} = aw'^{\mathcal{R}}$ and $v^{\mathcal{R}} = bv'^{\mathcal{R}}$, then G_1 and G_2 have rules $A_1 \to aB_1$ and $A_2 \to bB_2$, respectively, where $w'^{\mathcal{R}} \in L(G_1^{B_1})$ and $v'^{\mathcal{R}} \in L(G_2^{B_2})$. Therefore, G has either the rule $(A_1, A_2) \to a(B_1, B_2)$, if $b = \mathtt{u}$, or the rule $(A_1, A_2) \to (B_1, B_2)a$, if $b = \mathtt{d}$.
Assume, by induction hypothesis, that $h(w', v') \in L(G^{(B_1,B_2)})$. If $b = \mathtt{u}$, then (A_1, A_2) generates $ah(w', v') = h(w'a, v'\mathtt{u}) = h(w, v)$. If $b = \mathtt{d}$, then (A_1, A_2) generates $h(w', v')a = h(w'a, v'\mathtt{d}) = h(w, v)$. In either case, $s = h(w, v) \in L(G^{(A_1,A_2)})$.

Next, we show that any word $s \in L(G^{(A_1,A_2)})$ is also in $L(L(G_1^{A_1})^{\mathcal{R}}, L(G_2^{A_2})^{\mathcal{R}})$. Again, using induction on the length of s:

1. If $|s| = 0$, then $s = \varepsilon$ and G has a rule $(A_1, A_2) \to \varepsilon$. Therefore, G_1 and G_2 have rules $A_1 \to \varepsilon$ and $A_2 \to \varepsilon$, respectively, and $h(\varepsilon, \varepsilon) = \varepsilon \in L(L(G_1^{A_1})^{\mathcal{R}}, L(G_2^{A_2})^{\mathcal{R}})$.
2. If $|s| > 0$, then G has either a rule $(A_1, A_2) \to a(B_1, B_2)$ or a rule $(A_1, A_2) \to (B_1, B_2)a$, where $a \in \Sigma$. Consider the former case, and let $s = as'$, where $s' \in L(G^{(B_1,B_2)})$.
By induction hypothesis, assume that $s' \in L(L(G_1^{B_1})^{\mathcal{R}}, L(G_2^{B_2})^{\mathcal{R}})$. Then, there are words $w' \in L(G_1^{B_1})^{\mathcal{R}}$ and

$v' \in L(G_2^{B_2})^{\mathcal{R}}$ such that $s' = h(w', v')$. Also, rule $(A_1, A_2) \to a(B_1, B_2)$ implies that G_1 and G_2 have rules $A_1 \to aB_1$ and $A_2 \to \mathtt{u}B_2$, respectively, and then $w'a \in L(G_1^{A_1})^{\mathcal{R}}$ and $v'\mathtt{u} \in L(G_2^{A_2})^{\mathcal{R}}$. Thus, $h(w'a, v'\mathtt{u}) = ah(w', v') = s \in L(L(G_1^{A_1})^{\mathcal{R}}, L(G_2^{A_2})^{\mathcal{R}})$.
The case in which G has a rule $(A_1, A_2) \to (B_1, B_2)a$ is treated similarly, with $s = s'a$. We obtain that G_2 has a rule $A_2 \to \mathtt{d}B_2$, and then $v'\mathtt{d} \in L(G_2^{A_2})^{\mathcal{R}}$. Thus, $h(w'a, v'\mathtt{d}) = h(w', v')a = s \in L(L(G_1^{A_1})^{\mathcal{R}}, L(G_2^{A_2})^{\mathcal{R}})$. □

Appendix B. Proof of Claim in Theorem 2

Proof. Suppose that there is a word $w \in L_1$ such that $w = w_1\#w_2$ such that $|w_1| > N$, take any $v \in L_2$ such that $h(w, v) = u_1\#u_2$. Now let $v = v_1v_2$ with $|v_1| = |w_1|$. Since $|v_1| = |w_1| > N$, there are $x_1, x_2, y_1, y_2, z_1, z_2$ such that $w_1 = x_1y_1z_1$, $v_1 = x_2y_2z_2$, $|x_1| = |x_2|$, $|y_1| = |y_2| > 0$, $|z_1| = |z_2|$, and $x_1y_1^kz_1\#w_2 \in L_1$ and $x_2y_2^kz_2v_2 \in L_2$. It is straightforward to see that fact if we view the grammars that define L_1 and L_2 as finite automata, with number of states N_1 and N_2, respectively. Assign to every position in w_1 and v_1 the pair of states reached by the two automata when reading those words. Since $|w_1| = |v_1| > N$, then some pair of states appears at least twice, and the positions in w_1 and v_1 where the same pair has appeared delimit y_1 and y_2. The rest follows from the pumping lemma for regular languages.

Next, let $m = 2|w_2| + 1$ and take the corresponding words $w' = x_1y_1^mz_1\#w_2$ and $v' = x_2y_2^mz_2v_2$. Note that when folding w' under the control of v', before # is reached the prefix $x_1y_1^mz_1$ is folded into a word u' of length greater than $2|w_2| + 1$. Regardless of how the remaining part of w' is folded, the end result will be a word $u_1\#u_2$ where one of u_1 or u_2 contains the factor u' and therefore is longer than $2|w_2| + 1$ while the other word is no longer than $|w_2|$. This however contradicts the observation that follows immediately from the definition of L: every word of L has the form $u_1\#u_2$ such that $|u_1| \le 2|u_2|$ and $|u_2| \le 2|u_1|$. □

References

[1] C.M. Dobson, Protein folding and misfolding, Nature 426 (6968) (2003) 884–890, https://doi.org/10.1038/nature02261.
[2] L. Mahadevan, S. Rica, Self-organized origami, Science 307 (5716) (2005) 1740, https://doi.org/10.1126/science.1105169.
[3] S. Felton, M. Tolley, E. Demaine, D. Rus, R. Wood, A method for building self-folding machines, Science 345 (6197) (2014) 644–646, https://doi.org/10.1126/science.1252610.
[4] R.J. Lang, The science of origami, Phys. World 20 (2) (2007) 30–31, https://doi.org/10.1088/2058-7058/20/2/31.
[5] E.D. Demaine, J. O'Rourke, Geometric Folding Algorithms, Cambridge University Press, Cambridge, United Kingdom, 2007.
[6] D. Sburlan, Computing by folding, Int. J. Comput. Commun. Control 6 (4) (2011) 739–748, https://doi.org/10.15837/ijccc.2011.4.2106.
[7] L. Kari, G. Rozenberg, The many facets of natural computing, Commun. ACM 51 (10) (2008) 72–83, https://doi.org/10.1145/1400181.1400200.
[8] P.W.K. Rothemund, Folding DNA to create nanoscale shapes and patterns, Nature 440 (7082) (2006) 297–302, https://doi.org/10.1038/nature04586.
[9] G. Rozenberg, Gene assembly in ciliates: computing by folding and recombination, in: A. Salomon, D. Wood, S. Yu (Eds.), A Half-Century of Automata Theory, World Scientific, Singapore, 2001, pp. 93–130.
[10] J.C. Lucero, Pumping lemmas for classes of languages generated by folding systems, Nat. Comput. 20 (2) (2021) 321–327, https://doi.org/10.1007/s11047-019-09771-5.
[11] M. Sipser, Introduction to the Theory of Computation, 3rd edition, Cengage Learning, Boston, MA, 2013.
[12] A. Mateescu, A. Salomaa, Formal languages: an introduction and a synopsis, in: R. G., A. Salomaa (Eds.), Handbook of Formal Languages, Volume 1: Word, Language, Grammar, Springer, Berlin, Germany, 1997, pp. 1–39.
[13] J.-M. Autebert, J. Berstel, L. Boasson, Context-free languages and pushdown automata, in: A. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. 1, Word, Language, Grammar, Springer, Berlin, Germany, 1997, pp. 111–174.
[14] G. Horváth, B. Nagy, Pumping lemmas for linear and nonlinear context-free languages, Acta Univ. Sapientiae 2 (2) (2010) 194–209.
[15] M. Kutrib, A. Malcher, D. Wotschke, The Boolean closure of linear context-free languages, Acta Inform. 45 (3) (2008) 177–191, https://doi.org/10.1007/s00236-007-0068-6.
[16] H. Fernau, J.M. Sempere, Permutations and control sets for learning non-regular language families, in: A.L. Oliveira (Ed.), Grammatical Inference: Algorithms and Applications, Springer, Berlin, Germany, 2000, pp. 75–88.
[17] V. Amar, G. Putzolu, On a family of linear grammars, Inf. Control 7 (3) (1964) 283–291.