Defining Languages by Forbidding-Enforcing Systems

Daniela Genova

Department of Mathematics and Statistics, University of North Florida,
Jacksonville, FL 32224, USA
d.genova@unf.edu

Abstract. Motivated by biomolecular computing, forbidding-enforcing systems (fe-systems) were first used to define classes of languages (fe-families) based on boundary conditions. This paper presents a new model of fe-systems in which fe-systems define single languages (fe-languages) based on forbidden and enforced subwords. The paper characterizes well-known languages by fe-systems, investigates the relationship between fe-families and fe-languages, and describes how an fe-system can generate the solution to the k-colorability problem and model splicing.

Keywords: fe-languages, fe-families, formal languages, k-colorability problem, biomolecular computing, splicing.

1 Introduction

The rapid growth of biomolecular computing is interconnected with the quest for new ways to define computation. Many computational models were defined within the field of natural computing, e.g., self-assembly, splicing, membrane systems (see [9,11,12,14,16]). Most of these models are rooted in classical formal language theory. When computation is carried out by biomolecules, the nondeterministic behavior of molecules in a biochemical reaction inspires new non-deterministic ways of defining languages. Motivated by such non-determinism, and by abstracting molecules to strings and sets of molecules to languages, the authors of [2,4,15] introduced forbidding-enforcing systems (fe-systems) as language-defining systems, where "everything that is not forbidden is allowed", contrasting the determinism of grammars and automata where "everything that is not allowed is forbidden". They showed that fe-systems can generate solutions to computational problems such as SAT and Hamiltonian Path Problem, represent duplex DNA molecules and model splicing by an enzyme.

Forbidding-enforcing systems have been defined in the framework of membrane systems (see [1]), where the authors show that the additional restrictions imposed by membranes cause fe-systems to define yet other new classes of languages, and have been also defined to model self-assembly of graphs (see [5]).

A topological investigation of fe-families can be found in [8]. Such a study of formal languages comes to interest only with the introduction of fe-systems, since, as it is shown in [8], none of the Chomsky families of languages correspond to an open or a closed set in the defined metric space.

This paper uses the concept of forbidding and enforcing to present a new way of defining languages, where one forbidding-enforcing system defines one language instead of a family of languages. Such a motivation comes from laboratory setting, where after a "wet" computation is performed the solution will be a specific language of words (DNA molecules) over the DNA alphabet.

The paper is organized as follows. Section 2 defines fe-languages and their basic properties are discussed in Section 3. The relationship between fe-languages and fe-families is investigated in Section 4. Section 5 provides characterizations of well-known languages by fe-systems. Section 6 discusses forbidding through enforcing. Generating solutions to the k-colorability problem is discussed in Section 7 and Section 8 discusses modeling splicing by fe-systems.

2 Forbidding-Enforcing Systems

A finite set of symbols (*alphabet*) is denoted by A and the free monoid consisting of all words over A is denoted by A^*. A subset of A^* is called a *language*. The *length* of a word $w \in A^*$ is denoted by $|w|$ and A^m is the set of all words of length m, whereas $A^{\leqslant m}$ is the set of all words of length at most m. The empty word, denoted by λ has length 0. The language A^+ consists of all words over A with positive length.

The word $y \in A^*$ is a *factor* (*subword*) of $x \in A^*$, if there exist $s, t \in A^*$, such that $x = syt$. The set of subwords of a word x is denoted by $sub\,(x)$ and the set of subwords of a language L by $sub\,(L)$, where $sub\,(L) = \cup_{x \in L}\, sub\,(x)$.

This paper uses the definitions and notation for fe-families as in [4] and for maximal languages as introduced in [8]. For more details about properties of fe-systems defining fe-families of languages, the reader is referred to [4,2,3,15,6,8]. Where not explicitly stated, assume that an alphabet A is given.

2.1 Forbidding Systems, *f*-languages

This paper introduces an fe-systems model, in which one forbidding-enforcing system defines a single language as opposed to a family of languages. For the single language model, only the definition of a forbidding set (Definition 1) remains the same as in [4].

Definition 1. A *forbidding set* $\mathcal{F}$ is a family of finite nonempty subsets of A^+; each element of a forbidding set is called a *forbidder*.

Definition 2. A word w is *consistent with a forbidder* F, denoted by $w\, con\, F$, if $F \not\subseteq sub\,(w)$. A word w is *consistent with a forbidding set* $\mathcal{F}$ denoted by $w\, con\, \mathcal{F}$, if $w\, con\, F$ for all $F \in \mathcal{F}$. If w is not consistent with $\mathcal{F}$, the notation is $w\, ncon\, \mathcal{F}$. The language $L(\mathcal{F}) = \{w \mid w\, con\, \mathcal{F}\}$ is said to be *defined* by the forbidding set $\mathcal{F}$. A language L is a *forbidding language* or *f-language*, if there is a forbidding set $\mathcal{F}$ such that $L = L(\mathcal{F})$.

Example 1. Let $\mathcal{F} = \{\{ab, ba\}, \{aa, bb\}\}$. Then $L(\mathcal{F}) = \{a^n, b^n, ab^n, a^nb, ba^n, b^na \mid n \geq 0\}$.

Example 2. Let $A = \{a, b\}$ and $\mathcal{F} = \{\{b\}\}$. Then $L(\mathcal{F}) = a^*$.

Example 3. Let $A = \{a, b\}$ and $\mathcal{F} = \{\{bb\}\}$. Then $L(\mathcal{F})$ contains words where any two b's are separated by at least one a. Note that $a^* \subseteq L(\mathcal{F})$.

The first part of the following remark simply says that if nothing is forbidden, then everything is allowed.

Remark 1. 1. $L(\mathcal{F}) = A^*$ if and only if $\mathcal{F}$ is empty.
2. The empty word λ is in $L(\mathcal{F})$ for every $\mathcal{F}$.

2.2 Enforcing Systems, *e*-languages

Definition 3. An *enforcing set* $\mathcal{E}$ is a family of ordered pairs called *enforcers* (x, Y), such that $x \in A^*$ and $Y = \{y_1, \ldots, y_n\}$ where $y_i \in A^+$ for $i = 1, \ldots, n$, $x \in \mathit{sub}\,(y_i)$ and $x \neq y_i$ for every $y_i \in Y$. A word w *satisfies an enforcer* (x, Y) ($w\,\mathit{sat}\,(X, Y)$), if $w = uxv$ for some $u, v \in A^*$ implies that there exists $y_i \in Y$ and $u_1, u_2, v_1, v_2 \in A^*$ such that $y_i = u_2xv_2$ and $w = u_1u_2xv_2v_1$. A word w *satisfies an enforcing set* $\mathcal{E}$ ($w\,\mathit{sat}\,\mathcal{E}$), if w satisfies every enforcer in that set. If w does not satisfy $\mathcal{E}$, the notation is $w\,\mathit{nsat}\,\mathcal{E}$. For an enforcing set $\mathcal{E}$ the language of all words that satisfy it is denoted by $L(\mathcal{E})$. A language L is called an *e-language* if there exists an enforcing set $\mathcal{E}$ such that $L = L(\mathcal{E})$.

In the case that $x \notin \mathit{sub}\,(w)$, w is said to satisfy the enforcer trivially. We call enforcers in which $x = \lambda$ *brute*. In this case, a word from Y has to be a subword of w in order for w to satisfy the enforcer.

Remark 2. If $y \in Y$ then $y\,\mathit{sat}\,(x, Y)$.

Remark 3. $L(\mathcal{E}) = A^*$ if and only if $\mathcal{E} = \emptyset$.

An enforcer (x, Y) is called *strict* if $|Y| = 1$. The following example shows how a brute strict enforcer and an infinite set of strict enforcers may not be satisfied by any finite word.

Example 4. Let $\mathcal{E} = \{(\lambda, \{a\})\} \cup \{(a^i, \{a^{i+1}\}) \mid i \geq 1\}$. Then, $L(\mathcal{E}) = \emptyset$.

2.3 Forbidding-Enforcing Systems, fe-languages

Preserving the idea of a forbidding-enforcing system from [4], an analogous definition for a forbidding-enforcing language (fe-language) is presented.

Definition 4. A *forbidding-enforcing system* is an ordered pair $(\mathcal{F}, \mathcal{E})$, such that $\mathcal{F}$ is a forbidding set and $\mathcal{E}$ is an enforcing set. The language $L(\mathcal{F}, \mathcal{E})$ defined by this system consists of all words that are consistent with $\mathcal{F}$ and satisfy $\mathcal{E}$, i.e., $L(\mathcal{F}, \mathcal{E}) = L(\mathcal{F}) \cap L(\mathcal{E})$. A language L is called an *fe-language*, if there exists an fe-system $(\mathcal{F}, \mathcal{E})$, such that $L = L(\mathcal{F}, \mathcal{E})$.

Example 5. 1. Let $\mathcal{F} = \{\{ba\}\}$ and $\mathcal{E}_1 = \{(\lambda, \{a\})\} \cup \{(a^i, \{a^{i+1}, a^i b^i\}) \mid i \geq 1\}$. Then, $L_1 = L(\mathcal{F}, \mathcal{E}_1) = \{a^n b^m \mid n \leq m \text{ and } n, m \geq 1\}$.

2. Let $\mathcal{F} = \{\{ba\}\}$ and $\mathcal{E}_2 = \{(\lambda, \{b\})\} \cup \{(b^i, \{b^{i+1}, a^i b^i\}) \mid i \geq 1\}$. Then, $L_2 = L(\mathcal{F}, \mathcal{E}_2) = \{a^n b^m \mid n \geq m \text{ and } n, m \geq 1\}$.

3 Basic Properties of Forbidding-Enforcing Systems

The proposition below includes some immediate properties of forbidding and enforcing sets and the languages that they define. These properties are reminiscent of the ones proved for fe-families of languages [4,15].

Proposition 1. *Let $\mathcal{F}$ and $\mathcal{F}'$ be forbidding sets, $\mathcal{E}$ and $\mathcal{E}'$ be enforcing sets, and u and w be words.*

1. *If $u \in sub(w)$ and w con $\mathcal{F}$, then u con $\mathcal{F}$.*
2. *If $\mathcal{F}' \subseteq \mathcal{F}$, then $L(\mathcal{F}) \subseteq L(\mathcal{F}')$.*
3. *If $\mathcal{E}' \subseteq \mathcal{E}$, then $L(\mathcal{E}) \subseteq L(\mathcal{E}')$.*
4. *If $\mathcal{F}' \subseteq \mathcal{F}$ and $\mathcal{E}' \subseteq \mathcal{E}$, then $L(\mathcal{F}, \mathcal{E}) \subseteq L(\mathcal{F}', \mathcal{E}')$.*
5. *$L(\mathcal{F} \cup \mathcal{F}') = L(\mathcal{F}) \cap L(\mathcal{F}')$.*
6. *$L(\mathcal{E} \cup \mathcal{E}') = L(\mathcal{E}) \cap L(\mathcal{E}')$.*
7. *$L(\mathcal{F} \cup \mathcal{F}', \mathcal{E} \cup \mathcal{E}') = L(\mathcal{F}, \mathcal{E}) \cap L(\mathcal{F}', \mathcal{E}')$.*

Example 6. Consider the fe-systems from example 5. It follows from Property 7 above that $L = L_1 \cap L_2 = \{a^n b^n \mid n \geq 1\} = L(\mathcal{F}, \mathcal{E}_1 \cup \mathcal{E}_2)$.

4 Relationship between f-families and f-languages

Given an alphabet A consider a generating tree T_{A^*}. The root of the tree is λ. If u is a node in the tree, then ua is a child of u for all $a \in A$. Clearly, this tree contains all words in A^*. For more details on properties of T_{A^*} see [6]. It is obvious that for every language $L \subseteq A^*$, if the vertices that are not in L are removed from T_{A^*}, then the vertices of the resulting graph are precisely the words in L, but the resulting graph is not necessarily a tree. The resulting graphs for factorial languages, i.e. those for which $L = sub(L)$ however, are trees, as observed in the next section.

Consider again the forbidding set $\mathcal{F} = \{\{aa, bb\}, \{ab, ba\}\}$. It was discussed in [4,2,15,6,8], where it was used to define a family of languages. In [8] it was observed that $\mathcal{L}(\mathcal{F})$ has four maximal languages. Note that $L(\mathcal{F})$ from Example 1, is precisely the union of these four languages. The next theorem states that this is always the case. Figure 1 depicts the generating tree for $L(\mathcal{F})$.

Theorem 1. *Let $\mathcal{F}$ be a forbidding set. Let $\mathcal{M}(\mathcal{F})$ be the set of maximal languages for $\mathcal{F}$ and $L(\mathcal{F})$ the $\mathcal{F}$-language. Then, $L(\mathcal{F}) = \cup_{L \in \mathcal{M}(\mathcal{F})} L$.*

Proof. Let $\mathcal{F}$ be given. Consider $w \in L(\mathcal{F})$. Since $F \not\subseteq sub(w)$ for every $F \in \mathcal{F}$, w is in some language $K \in \mathcal{L}(\mathcal{F})$. Hence, w is in some maximal language L in $\mathcal{M}(\mathcal{F})$. It follows that $L(\mathcal{F}) \subseteq \cup_{L \in \mathcal{M}(\mathcal{F})} L$. Conversely, let $w \in \cup_{L \in \mathcal{M}(\mathcal{F})} L$. Then, $w \in L$ for some $L \in \mathcal{M}(\mathcal{F})$. It follows that for every $F \in \mathcal{F}$, $F \not\subseteq sub(w)$. Thus, $w \in L(\mathcal{F})$. □

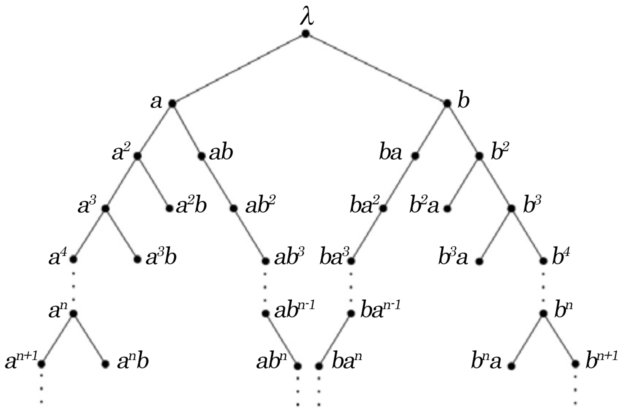

Fig. 1. The tree representing $L(\mathfrak{F})$ from Example 1

5 Characterizing Formal Languages by Forbidding-Enforcing Systems

This section presents some characterizations of formal languages by fe-systems. It begins with the straightforward characterization of local languages.

Proposition 2. *L is local if and only if there is $\mathfrak{F} = \{\{v_1\}, \ldots, \{v_n\}\}$, such that $L = L(\mathfrak{F})$.*

Proof. A language L is local if and only if there exists a finite set of words $H = \{v_1, \ldots, v_n\}$ such that $L = A^* \backslash A^* H A^*$, which holds if and only if no word from L has a subword from H, i.e., if and only if $L = L(\mathfrak{F})$, where $\mathfrak{F} = \{\{v_1\}, \ldots, \{v_n\}\}$. □

The following proposition states that not all languages are f-languages.

Proposition 3. *There exists a language that is not a forbidding language.*

Proof. Let $L = \{a, ba\}$. Observe that the word b is forbidden if and only if $\{b\}$ is a forbidder, in which case ba is forbidden. Thus, L is not an f-language. □

This leads to the conclusion that if a factor (subword) u of a word x from a language L is such that $u \notin L$, then forbidding u leads to forbidding x which means that L cannot be an f-language. Thus, a finite language is not necessarily an f-language, but it is an fe-language, as shown later in this section.

The following lemma states that only factorial languages can be f-languages.

Lemma 1. *Let $\mathfrak{F}$ be a forbidding set. Then, $L(\mathfrak{F})$ is factorial.*

Proof. Let $\mathfrak{F}$ be given and $w \in L(\mathfrak{F})$. Let $u \in sub(w)$. If $F \subseteq sub(u)$ for some $F \in \mathfrak{F}$, then $F \subseteq sub(w)$, which contradicts our assumption. Hence $u \in L(\mathfrak{F})$ and $L(\mathfrak{F})$ is factorial. □

The converse question, whether every factorial language is an f-language, is investigated next. In the case of f-families, it was shown in [8] that even though every maximal language in an f-family is factorial, not every family of languages for which the maximal languages are factorial is an f-family.

Remark 4. Every factorial language L can be represented by a directed tree T_L where the labels of the vertices are exactly the words in L.

Given a factorial language L, consider the tree T_L obtained from T_{A^*} as follows. For each $w \notin L$ let u be the longest prefix of w, such that $u \in L$ and let $w = uav$ where $a \in A$ and $u, v \in A^*$. Then, the tree T_L does not contain vertex ua along with all paths that begin at ua, i.e., the entire branch rooted at ua is removed from T_{A^*}. It is easy to see that the labels of the vertices of T_L are precisely the words in L.

Lemma 2. *Let L be factorial. Then, L is an f-language.*

Proof. Let L be a factorial language. From the tree T_L construct $\mathcal{F} = \{\{ua\} \mid u \in V(T_L), ua \notin V(T_L), a \in A\}$. Then, $w \in L(\mathcal{F})$ if and only if $x \notin sub(w)$ for all $\{x\} \in \mathcal{F}$ if and only if $w \in V(T_L)$, i.e., $w \in L$. Consequently, $L = L(\mathcal{F})$. □

The next theorem follows from Lemmas 1 and 2 above.

Theorem 2. *A language is factorial if and only if it is an f-language.*

Some well-known languages can be characterized by enforcing systems only. Example 4 shows that an enforcing set that generates only infinite words, defines the empty language. Hence, the empty language is an e-language.

Proposition 4. *There exists a non-semilinear language that is an e-language.*

Proof. Let $A = \{a\}$ and $L = \{a^{2^n} \mid n \geq 0\}$. Then, the enforcing set $\mathcal{E} = \{(\lambda, \{a, aa\})\} \cup \{(a^{2^i+1}, \{a^{2^{i+1}}\}) \mid i \geq 1\}$ defines L, i.e., $L = L(\mathcal{E})$. □

Proposition 5. *There exists a non-regular linear language that is an fe-language.*

Proof. Let $\mathcal{F} = \{\{ba\}\}$, $\mathcal{E} = \{(\lambda, \{a\}), (\lambda, \{b\})\} \cup \{(a^i, \{a^{i+1}, a^ib^i\}) \mid i \geq 1\} \cup \{(b^i, \{b^{i+1}, a^ib^i\}) \mid i \geq 1\}$. Then, as noted in Example 6, $L = \{a^nb^n \mid n \geq 1\} = L(\mathcal{F}, \mathcal{E})$. □

Proposition 6. *There exists a non-linear context-free language that is an fe-language.*

Proof. Let $L_1 = \{a^nb^n \mid n \geq 1\}$ and consider $L = L_1L_1$. Then, $L = L(\mathcal{F}, \mathcal{E})$ where $\mathcal{F} = \{\{ba^ib^ia\} \mid i \geq 1\}$ and $\mathcal{E} = \{(\lambda, \{a, b\})\} \cup \{(a^i, \{a^{i+1}, ba^ib^i, a^ib^ia\}) \mid i \geq 1\} \cup \{(b^i, \{b^{i+1}, ba^ib^i, a^ib^ia\}) \mid i \geq 1\} \cup \{(b^ja^ib^i, \{b^{j+1}a^ib^i, a^jb^ja^ib^i\}) \mid j \geq 1, i \geq 1\} \cup \{(a^ib^ia^j, \{a^ib^ia^{j+1}, a^ib^ia^jb^j\}) \mid j \geq 1, i \geq 1\}$. □

Proposition 7. *There exists a non-context-free language that is an fe-language.*

Proof. Let $\mathcal{F} = \{\{ba\}, \{ca\}, \{ac\}, \{cb\}\}$ and $\mathcal{E} = \{(\lambda, \{a, b, c\})\} \cup \{(a^i, \{a^{i+1}, a^i b^i c^i\}), (b^i, \{b^{i+1}, a^i b^i c^i\}), (c^i, \{c^{i+1}, a^i b^i c^i\}) \mid i \geq 1\}$. Then, $L(\mathcal{F}, \mathcal{E}) = \{a^n b^n c^n \mid n \geq 1\}$. □

Example 2 and Propositions 5, 6 and 7 prove the following statement. Assume that FIN, REG, LIN, CF, and CS denote the classes of finite, regular, linear, context-free, and context-sensitive languages respectively.

Theorem 3. *For every $X \in \{REG - FIN, LIN - REG, CF - LIN, CS - CF\}$ there exists $L \in X$ such that L is an fe-language.*

We conclude this section with an fe-characterization of finite languages.

Proposition 8. *Every finite language is an fe-language.*

Proof. Let $L = \{x_1, x_2, \ldots, x_n\}$ be a finite language with $m = max\{|w| \mid w \in L\}$. Construct $\mathcal{F} = \{\{w\} \mid w \in A^{m+1}\}$ and $\mathcal{E} = \{(w, \{u \mid u \in L \text{ and } w \in sub(u)\} \cup \{v_w \mid v_w \in A^{m+1} \text{ and } w \in sub(v_w)\}) \mid w \notin L \text{ and } w \in A^{\leqslant m}\}$. We show that $L = L(\mathcal{F}, \mathcal{E})$. Assume that $w \in L$. Obviously, $w\, con\, \mathcal{F}$. If there is a (x, Y) such that $x \in sub(w)$, then $w \in Y$. Hence, $w\, sat\, \mathcal{E}$. Consequently, $L \subseteq L(\mathcal{F}, \mathcal{E})$. Conversely, if $w \notin L$, then either $|w| \geq m + 1$ which implies that $w\, ncon\, \mathcal{F}$ or $|w| \leq m$ which implies that there exists an enforcer $(w, Y) \in \mathcal{E}$ and therefore, $w\, nsat\, \mathcal{E}$. In either case, $w \notin L(\mathcal{F}, \mathcal{E})$. Hence, $L(\mathcal{F}, \mathcal{E}) \subseteq L$. Consequently, $L = L(\mathcal{F}, \mathcal{E})$. □

Using the proof of the above proposition, we now construct an fe-system that defines the non-forbidding language $L = \{a, ba\}$ from the proof of Proposition 3. The fe-system $(\mathcal{F}, \mathcal{E})$ where $\mathcal{F} = \{\{aaa\}, \{aab\}, \ldots, \{bbb\}\}$ and $\mathcal{E} = \{(\lambda, \{a, ba, aaa\}), (b, \{ba, bbb\}), (aa, \{aaa\}), (ab, \{aba\}), (bb, \{bba\})\}$ is such that $L = L(\mathcal{F}, \mathcal{E})$.

6 Forbidding through Enforcing

The concept of minimal connect introduced below is used to replace forbidders by enforcers.

Definition 5. Let F be a finite set of words. A word y is called a *connect* of F if $F \subseteq sub(y)$. A word x is called a *minimal connect* of F if x is a connect of F and for every connect y of F it holds that $y \in sub(x)$ implies $y = x$. The set of minimal connects of F is denoted by $C_{min}(F)$.

Theorem 4. *For every forbidding set $\mathcal{F}$ there exists an enforcing set $\mathcal{E}$, such that $L(\mathcal{F}) = L(\mathcal{E})$.*

Proof. Let $\mathcal{F}$ be a forbidding set and $a \in A$ be some symbol. Given $F \in \mathcal{F}$, consider $C_{min}(F)$ and for every $u \in C_{min}(F)$ define $\mathcal{E}_{F_u} = \{(u, \{ua\}), (ua, \{uaa\}), \ldots, (ua^n, \{ua^{n+1}\}), \ldots\}$. Let $\mathcal{E}_F = \cup_{u \in C_{min}(F)} \mathcal{E}_{F_u}$ and $\mathcal{E} = \cup_{F \in \mathcal{F}} \mathcal{E}_F$. If $w\, nsat\, \mathcal{E}$, then there is an enforcer $(ua^i, \{ua^{i+1}\})$ which is not satisfied, which implies that

$ua^i \in sub(w)$ and one of its copies in w is not enclosed in ua^{i+1}. Hence, $u \in sub(w)$ and since $u \in C_{min}(F)$ for some $F \in \mathcal{F}$, $w\, ncon\, \mathcal{F}$. Conversely, if $w\, ncon\, \mathcal{F}$ there exists an $F \in \mathcal{F}$ and $u \in C_{min}(F)$ such that $u \in sub(w)$. Since w is a finite word, it cannot satisfy the enforcing set $\mathcal{E}_{F_u}$. Hence $w\, nsat\, \mathcal{E}$. □

Remark 5. Note that the above result does not necessarily make forbidding sets obsolete. Even though every forbidding set may be replaced by an enforcing set, replacing a finite forbidding set with an infinite number of enforcers may be undesirable.

7 *k*-colorability Problem

Forbidding-enforcing systems can generate solutions to computational problems. In [7] a more general, categorical description of the model was presented and a solution to the k-colorability problem defined by an fe-system in the category of sets was presented to illustrate information processing capabilities of fe-systems. This section shows how a solution to this well-known NP-complete problem can be described by fe-systems defining languages.

The k-colorability problem asks whether given a graph and a finite set of k colors it is possible to assign one color to each vertex in such a way that adjacent vertices have distinct colors. Such assignment of colors is called a k-coloring.

Let $G = (V, E)$ be a graph with n vertices, i.e. $V = \{v_1, v_2, \ldots, v_n\}$ and C be a set of k colors, i.e. $C = \{c_1, c_2, \ldots, c_k\}$. A k-coloring can be viewed as a word over the alphabet $A = V \cup C$ with specific properties.

fe-system construction. A combination of brute enforcing and forbidding is used to design an fe-system that corresponds to assigning exactly one color to a vertex. The enforcing set $\mathcal{E} = \{(\lambda, \{vc_1, vc_2, \ldots, vc_k\}) \mid v \in V\}$ ensures that every vertex is assigned at least one color. The forbidding set $\mathcal{F} = \{\{vc, vc'\} \mid v \in V$ and $c, c' \in C$ with $c \neq c'\}$ allows only these vertices that are assigned at most one color. Thus, every word in $L(\mathcal{F}, \mathcal{E})$ contains every vertex colored in exactly one color. Also, no two adjacent vertices should be colored the same. This is obtained from the forbidding set $\mathcal{F}' = \{\{uc, vc\} \mid \{u, v\} \in E$ and $c \in C\}$.

Following the above notation and construction we have the next result.

Theorem 5. *G is k-colorable if and only if $L(\mathcal{F} \cup \mathcal{F}', \mathcal{E}) \neq \emptyset$. Furthermore, any word $w \in L(\mathcal{F} \cup \mathcal{F}', \mathcal{E})$ represents exactly one k-coloring of G.*

Proof. Assume that G is k-colorable. Then, there exists a k-coloring c. Construct the word $w = v_1 c_{i_1} v_2 c_{i_2} \ldots v_n c_{i_n}$ where $c_{i_j} \in C$ and is the color c_i from the coloring c which corresponds to vertex v_j. Then, by construction of the fe-system $w \in L(\mathcal{F} \cup \mathcal{F}', \mathcal{E})$. Conversely, if the fe-language is not empty, there is $w \in L(\mathcal{F} \cup \mathcal{F}', \mathcal{E})$ which is a word over A. Since $w\, sat\, \mathcal{E}$ there is at least one $vc_j \in sub(w)$ for every $v \in V$ and some $c_j \in C$. Furthermore, no adjacent vertex to v can be colored by the same color because $w\, con\, \mathcal{F}'$ and no subword of w is of the form

vc_i where $j \neq i$ since $w\, con\, \mathcal{F}$. Observe that vc_j may have many copies in w, but w represents exactly one k-coloring of G. $\square$

8 Modeling Splicing

In [2,4] the authors show how fe-systems can define fe-families that describe cutting by an enzyme, recombination, and can model splicing. For details about the computational model of splicing systems, the reader is referred to [9,10,13,14]. This section shows how fe-systems defining fe-languages can be used to model these operations.

Cutting by an enzyme and recombination. Let $xuvy$ be a string modeling a double stranded DNA molecule where uv models the restriction site for some restriction enzyme, which upon cutting between u and v produces either two sticky overhangs or two blunt ends. Then, the language $L(\mathcal{F})$, where $\mathcal{F} = \{\{uv\}\}$ contains all molecules resulting after the cutting by this enzyme, as well as all other molecules that do not have the uv restriction site. Consequently, starting from $L(\mathcal{F})$, as a next step, one can model recombination by the enforcing set $\mathcal{E} = \{(u, \{uv\} \cup \{ua \mid a \in A\}), (v, \{uv\} \cup \{av \mid a \in A\})\}$. This contrasts the fe-families model presented in [2,4] where both overhangs and ligated strands are present in the fe-family. Another difference is that the above $\mathcal{E}$ is finite, whereas the enforcing set for the fe-families is infinite.

Splicing. Consider the words $x_1u_1v_1y_1$ and $x_2u_2v_2y_2$ and the splicing rule $r = (u_1, v_1; u_2, v_2)$, according to which the two words can be spliced to obtain the word $x_1u_1v_2y_2$, and by symmetry, the word $x_2u_2v_1y_1$. Observe that the enforcing set $\mathcal{E}_r = \{(u_1, \{u_1v_1, u_1v_2\} \cup \{u_1a \mid a \in A\}), (u_2, \{u_2v_1, u_2v_2\} \cup \{u_2a \mid a \in A\}), (v_1, \{u_1v_1, u_2v_1\} \cup \{av_1 \mid a \in A\}), (v_2, \{u_1v_2, u_2v_2\} \cup \{av_2 \mid a \in A\})\}$ models splicing for the splicing rule r and words over the alphabet A. This system is finite as opposed to the fe-family model which uses infinitely many enforcers to model one splicing rule.

9 Concluding Remarks

Similar to grammars and automata, forbidding-enforcing systems are structures that define languages. Unlike grammars and automata where a language is defined by generating or accepting every word symbol by symbol, forbidding-enforcing systems define a language by imposing restrictions on its subwords. Use of forbidding-enforcing systems, in which one fe-system defines a single language as opposed to a family of languages presents a new way of defining languages.

It was shown that the Chomsky classes of languages have representatives that can be defined by fe-systems. Characterizations of finite languages, local languages, and factorial languages by fe-systems were presented. It will be interesting to investigate which other classes of languages can be defined by fe-systems. Further applications of fe-systems to biomolecular computing should be investigated both theoretically and experimentally.

Acknowledgement

The author thanks the anonymous referee whose comments and suggestions improved a previous version of this paper. This work has been partially supported by a UNF Faculty Development Scholarship Grant.

References

1. Cavaliere, M., Jonoska, N.: Forbidding and enforcing in membrane computing. Natural Computing 2, 215–228 (2003)
2. Ehrenfeucht, A., Hoogeboom, H.J., Rozenberg, G., van Vugt, N.: Forbidding and enforcing. In: Winfree, E., Gifford, D.K. (eds.) DNA Based Computers V. AMS DIMACS, Providence, RI, vol. 54, pp. 195–206 (2001)
3. Ehrenfeucht, A., Hoogeboom, H.J., Rozenberg, G., van Vugt, N.: Sequences of languages in forbidding-enforcing families. Soft Computing 5(2), 121–125 (2001)
4. Ehrenfeucht, A., Rozenberg, G.: Forbidding-enforcing systems. Theoretical Computer Science 292, 611–638 (2003)
5. Franco, G., Jonoska, N.: Forbidding and Enforcing Conditions in DNA Self-assembly of Graphs. Nanotechnology: Science and Computation, Natural Computing Series, Part I 105–118 (2006)
6. Genova, D.: Forbidding and Enforcing of Formal Languages, Graphs and Partially Ordered Sets, PhD Thesis, University of South Florida (2007)
7. Genova, D., Jonoska, N.: Defining structures through forbidding and enforcing constraints. Physica B: Condensed Matter 394(2), 306–310 (2007)
8. Genova, D., Jonoska, N.: Topological Properties of Forbidding-Enforcing Systems. Journal of Automata, Languages and Combinatorics 11(4), 375–397 (2006)
9. Head, T.: Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. Bull. Math. Biology 49(6), 737–759 (1987)
10. Head, T., Păun, G., Pixton, D.: Language theory and molecular genetics: generative mechanisms suggested by DNA recombination. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 2, pp. 295–360. Springer, Berlin (1996)
11. Jonoska, N., Sa-Ardyen, P., Seeman, N.C.: Computation by self-assembly of DNA graphs. Journal of Genetic Programming And Evolvable Machines 4(2), 123–138 (2003)
12. Păun, G.: Membrane Computing. An Introduction. Springer, Berlin (2002)
13. Păun, G., Rozenberg, G., Salomaa, A.: Computing by splicing. Theoretical Computer Science 168, 321–336 (1996)
14. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing, new computing paradigms. Springer, Heidelberg (1998)
15. van Vugt, N.: Models of Molecular Computing, PhD thesis, Leiden University (2002)
16. Winfree, E., Yang, X., Seeman, N.C.: Universal Computation via Self-assembly of DNA: Some Theory and Experiments. In: Landweber, L., Baum, E. (eds.) DNA computers II. AMS DIMACS series, vol. 44, pp. 191–198 (1998)