DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume **54**, 2000

Forbidding and Enforcing

Andrzej Ehrenfeucht, Hendrik Jan Hoogeboom, Grzegorz Rozenberg,
and Nikè van Vugt

ABSTRACT. DNA molecules and various operations on them can be conveniently expressed as strings and operations on strings. Hence, many models of DNA computation have been formulated within formal language theory.

We propose a novel kind of model, which is based on two types of boundary conditions : forbidding and enforcing. *Forbidding conditions* say that a 'conflicting' group of components may not be present in a system, *enforcing conditions* say that if a certain group of molecules is present in the system, then some other molecules will eventually be present in the system.

Such forbidding-enforcing systems are 'tolerant' in describing results of (molecular) computations : one system describes the whole family of outcomes all of which obey the forbidding and enforcing constraints of the system, specifying a possibly infinite family of languages. This should be contrasted with standard formal language theory (grammars and automata) where a grammar specifies one language of all words that can be generated.

We illustrate the use of forbidding-enforcing systems for the description of the structure of DNA molecules, the description of splicing systems, and the description of the satisfiability problem.

Next to standard issues such as normal forms, we investigate two central issues : finiteness and the structure of computation.

1. Introduction

The research on DNA computing has lead to many novel models of computation, see for example [**HPP97**], [**Amo97**], [**Win98**]. Since DNA molecules can be expressed in a very natural way as strings (or double strings), and various operations on DNA can be expressed then as operations on strings, many of these models have been formulated within formal language theory. In a typical model of this sort (see, e.g., splicing systems), after the initial set of molecules (strings) and the operations on them (productions) are specified, one gets a rewriting system which defines a single language. This language represents the set of all molecules that can be obtained from the initial ones using the given operations.

We propose here a different kind of model, which is based on boundary conditions. We propose a description of molecular systems based on two types of boundary conditions : forbidding and enforcing. *Forbidding conditions* say that if a certain group of components is present in the system, then the system will 'die' (e.g., it will lose its functionality) – hence such a combination of components is

2000 *Mathematics Subject Classification.* Primary 68Q05, 68Q45.

forbidden. *Enforcing conditions* say that if a certain group of molecules is present in the system, then (as the result of a molecular reaction) some other molecules will eventually be present in the system. Hence the evolution of a system described by forbidding conditions $\mathcal{F}$ and enforcing conditions $\mathcal{E}$ will proceed according to the reactions described by $\mathcal{E}$ but restricted in such a way that none of the forbidding conditions from $\mathcal{F}$ will be created. Such forbidding-enforcing systems, *fe systems* for short, are much more 'tolerant' in describing results of (molecular) computations : one fe system describes the whole family of outcomes all of which obey the forbidding and enforcing constraints of the system. Thus in case when we model the molecules by strings (as we do in this paper), one fe system specifies a possibly infinite family of languages. A language belongs to this family if and only if it is consistent with the forbidding conditions and it satisfies the enforcing conditions – nothing else is required from the language. As a matter of fact, fe systems follow the rule "everything that is not forbidden is allowed", while standard formal language theory (grammars and automata) follows the dual rule "everything that is not allowed is forbidden". Hence fe systems lead in our opinion to models of computation novel from both the (modeling of) molecular systems and computation theory point of view.

This paper is organized as follows. First we introduce and investigate forbidding sets and enforcing sets, and then we combine the two and introduce forbidding-enforcing systems, which constitute our model of molecular systems. Next to standard issues such as, e.g., normal forms, we investigate two central issues.
(1) *Finiteness.* This issue is always important from the 'real world' point of view. We investigate many facets of finiteness, and we also discuss the very different role that finiteness plays in our model when compared to standard grammatical models.
(2) *Structure of computation.* Although fe systems are defined by boundary conditions, they turn out to be intrinsically computational. Each fe system, when transformed into a suitable normal form, can be *completely* represented by a tree which contains the information about the family of languages defined by the system, and also about all possible evolving computations of the system.

Finally we want to point out that we have not included the proofs of our results because the size of the paper would then become too big for this volume.

2. Preliminaries

We denote the set of (positive) natural numbers by $\mathbb{N}$, and the empty set by $\varnothing$.

In this paper we consider *non-empty* words only. For the ease of notation, we will fix one alphabet Σ, i.e., every word, language or family of languages is over Σ, unless clear otherwise. The set of all non-empty words over Σ is denoted by Σ^+ (occasionally we will also use the notation Σ^* which denotes the set of all words over Σ, including the empty word).

The length of a word w is denoted by $|w|$. For a language K and an integer $n \geq 1$, $K|_{\leq n} = \{w \in K \mid |w| \leq n\}$. A sequence of languages $K_1, K_2, \ldots$ is called *ascending* if $K_1 \subseteq K_2 \subseteq \ldots$. A sequence of languages $K_1, K_2, \ldots$ *converges* to a language K, denoted $\langle K_n \rangle_{n \in \mathbb{N}} \to K$, if, for each $\ell \geq 1$, there is an $m \geq 1$ such that $K_n|_{\leq \ell} = K|_{\leq \ell}$ for every $n \geq m$.

A word $x \in \Sigma^+$ is a *subword* of a word $y \in \Sigma^+$, denoted $x \underline{\text{sub}}\ y$, if $y = uxv$ for some $u, v \in \Sigma^*$. The set of subwords of a word x is denoted by $\underline{\text{sub}}\,(x)$. The set of

subwords of a language K, denoted $\underline{\text{sub}}(K)$, equals $\bigcup_{x\in K} \underline{\text{sub}}(x)$. A language K is called *subword free* if, for all $x, y \in K$, $x \underline{\text{sub}}\ y$ implies $x = y$.

3. Forbidding sets

In this section we formalize the first sort of boundary conditions, viz., forbidding conditions. To this aim we introduce the notion of a forbidding set.

3.1. Definitions and basic properties.

DEFINITION 1. A *forbidding set* is a (possibly infinite) family of finite languages; these finite languages are called *forbidders.*

A language K is said to be *consistent* with a forbidder F, denoted $K \underline{\text{con}}\ F$, if $F \not\subseteq \underline{\text{sub}}(K)$. A language K is consistent with a forbidding set $\mathcal{F}$, denoted $K \underline{\text{con}}\ \mathcal{F}$, if K is consistent with every forbidder in $\mathcal{F}$. □

EXAMPLE 2. Consider the forbidding set $\mathcal{F} = \{\{ab, ba\}, \{aa, bb\}\}$ and alphabet $\{a, b\}$. Then it is easily seen that $K \underline{\text{con}}\ \mathcal{F}$ iff $K \subseteq K_i$ for some $i \in \{1, 2, 3, 4\}$, where $K_1 = ab^* \cup b^+$, $K_2 = a^*b \cup a^+$, $K_3 = ba^* \cup a^+$, $K_4 = b^*a \cup b^+$. □

For a forbidding set $\mathcal{F}$ we define the family of ($\mathcal{F}$-)consistent languages $\mathcal{L}(\mathcal{F}) = \{K \mid K \underline{\text{con}}\ \mathcal{F}\}$, and the family of finite consistent languages $\mathcal{L}_{\text{fin}}(\mathcal{F}) = \{K \mid K \text{ is finite and } K \in \mathcal{L}(\mathcal{F})\}$.

THEOREM 3. *Let $\mathcal{F}$ be a forbidding set and K a language.*
(1) K $\underline{\text{con}}$ $\mathcal{F}$ implies $\underline{\text{sub}}(K)$ $\underline{\text{con}}$ $\mathcal{F}$
(2) If $K' \subseteq K$ and K $\underline{\text{con}}$ $\mathcal{F}$, then K' $\underline{\text{con}}$ $\mathcal{F}$
(3) If $K_1, K_2, \ldots$ is an ascending sequence of languages with K_i $\underline{\text{con}}$ $\mathcal{F}$ for all $i \geq 1$, then $(\bigcup_{i\geq 1} K_i)$ $\underline{\text{con}}$ $\mathcal{F}$

In the third property above, we infer the consistency of the language K from a converging sequence of fragments of K. This property follows from our assumption that forbidders are finite sets.

3.2. A useful minimal normal form. In this subsection we consider three technically useful normal forms for forbidding sets. We use the following obvious notion of equivalence : two forbidding sets $\mathcal{F}_1$ and $\mathcal{F}_2$ are *equivalent* if $\mathcal{L}(\mathcal{F}_1) = \mathcal{L}(\mathcal{F}_2)$.

Let $\mathcal{F}$ be a forbidding set, and let $F_1, F_2 \in \mathcal{F}$ with $F_1 \neq F_2$. We say that F_1 and F_2 are *subword incomparable* if neither $\underline{\text{sub}}(F_1) \subseteq \underline{\text{sub}}(F_2)$, nor $\underline{\text{sub}}(F_2) \subseteq \underline{\text{sub}}(F_1)$, and we call $\mathcal{F}$ subword incomparable if each pair of distinct forbidders $F_1, F_2 \in \mathcal{F}$ is subword incomparable. This is a normal form : for every forbidding set $\mathcal{F}$ there exists an equivalent subword incomparable forbidding set $\mathcal{F}'$. Note that $\mathcal{F}'$ is not always unique : for instance, constructing a subword incomparable equivalent of $\{\{a, ab\}, \{b, ab\}\}$ can yield both $\{\{a, ab\}\}$ and $\{\{b, ab\}\}$, since $\underline{\text{sub}}(\{a, ab\}) = \underline{\text{sub}}(\{b, ab\})$.

We call a forbidding set *subword free* if all its forbidders are subword free. For every forbidding set there exists an equivalent subword free forbidding set.

A forbidding set $\mathcal{F}$ is in *minimal normal form* if $\mathcal{F}$ is subword free and subword incomparable.

THEOREM 4. *For every forbidding set $\mathcal{F}$ there exists an equivalent forbidding set $\mathcal{F}'$ in minimal normal form. $\mathcal{F}'$ is unique.*

EXAMPLE 5. The (fortunately finite) equivalent in minimal normal form of the infinite forbidding set $\mathcal{F} = \{\{a^i, b^i, a^ib^i\} \mid i \geq 1\}$ is $\mathcal{F}' = \{\{ab\}\}$, since $\{a^i, b^i, a^ib^i\} \not\subseteq \underline{\text{sub}}(K)$ for all $i \geq 1$ iff $\{a, b, ab\} \not\subseteq \underline{\text{sub}}(K)$ iff $\{ab\} \not\subseteq \underline{\text{sub}}(K)$. □

A forbidding set in minimal normal form is indeed minimal, or 'redundancy free', in the sense that removing one of its forbidders or even one element from one forbidder yields a forbidding set that is not equivalent to the original one.

3.3. Finiteness. We turn now to the issue of finiteness for forbidding sets. First we observe that we cannot restrict ourselves to finite forbidding sets. On the other hand, the finite consistent languages characterize all consistent languages of a forbidding set.

For some specific infinite forbidding sets it is possible to find an equivalent *finite* forbidding set : $\mathcal{F}_2 = \{\{ab\}\}$ clearly is a finite 'summary' of $\mathcal{F}_1 = \{\{ab\}, \{a^2b^2\}, \{a^3b^3\}, \dots\}$. Unfortunately this is not always the case, which is seen as follows. For a finite forbidding set $\mathcal{F}$, let $\ell = \max\{|w| \mid w \in \bigcup_{F\in\mathcal{F}} F\}$. Now, if for two words x and y it is the case that $\underline{\text{sub}}(x)|_{\leq\ell} = \underline{\text{sub}}(y)|_{\leq\ell}$, then $\mathcal{F}$ cannot distinguish between x and y, which means that $\{x\} \in \mathcal{L}(\mathcal{F})$ if and only if $\{y\} \in \mathcal{L}(\mathcal{F})$.

Now consider $\mathcal{F}_{\text{even}} = \{\{ab^2a\}, \{ab^4a\}, \{ab^6a\}, \dots\}$. For each ℓ, the words $x = ab^\ell a$ and $y = ab^{\ell+1}a$ differ only on subwords of length greater than ℓ. Clearly $\{x\} \in \mathcal{L}(\mathcal{F}_{\text{even}})$ if and only if $\{y\} \notin \mathcal{L}(\mathcal{F}_{\text{even}})$. Hence $\mathcal{L}(\mathcal{F}_{\text{even}}) \neq \mathcal{L}(\mathcal{F})$ for all finite $\mathcal{F}$.

Theorem 3 (2) and (3) together yield the following result, which states that consistence of certain specific finite parts of a language K is necessary and sufficient to ensure the consistence of K itself.

LEMMA 6. *K $\underline{con}$ $\mathcal{F}$ if and only if $K|_{\leq i}$ $\underline{con}$ $\mathcal{F}$ for every $i \geq 1$.*

As a matter of fact, finite languages are central elements of every family of consistent languages, as demonstrated by the following result.

THEOREM 7. *For all forbidding sets $\mathcal{F}_1$ and $\mathcal{F}_2$ the following holds : $\mathcal{L}(\mathcal{F}_1) = \mathcal{L}(\mathcal{F}_2)$ if and only if $\mathcal{L}_{\text{fin}}(\mathcal{F}_1) = \mathcal{L}_{\text{fin}}(\mathcal{F}_2)$.*

4. Enforcing sets

In this section we formalize the second sort of boundary conditions, viz., enforcing conditions. To this aim we introduce the notion of enforcing sets.

4.1. Definitions and basic properties.

DEFINITION 8. An *enforcing set* is a (possibly infinite) family of ordered pairs (X, Y), where X and Y are finite languages with $Y \neq \varnothing$; such a pair (X, Y) is called an *enforcer*.

A language K *satisfies* an enforcer (X, Y), denoted K $\underline{\text{sat}}$ (X, Y), if $X \subseteq K$ implies $Y \cap K \neq \varnothing$. A language K satisfies an enforcing set $\mathcal{E}$, denoted K $\underline{\text{sat}}$ $\mathcal{E}$, if K satisfies every enforcer in $\mathcal{E}$. □

EXAMPLE 9. The family $\mathcal{E} = \{(\{u, v\}, \{uv, vu\}) \mid u, v \in \Sigma^+\}$ is an enforcing set. If $K \subseteq \Sigma^+$ satisfies $\mathcal{E}$, then K is closed under 'weak catenation' : for any words $u, v \in K$ at least one of the words uv, vu is in K. Note that there are infinitely many languages satisfying $\mathcal{E}$, each resulting from a different 'implementation' of the weak catenation. □

Note the essential use in the above example of the non-determinism of enforcers. Given an enforcer (X, Y) and a language K, if the set X of *premises* is included in K, then at least one element of the set Y of *consequences* will be included in K. Thus allowing Y to include more than one element accounts for the non-determinism. In the world of molecular reactions the result of molecules from X reacting together may depend quite essentially on all kinds of conditions under which the reaction takes place (e.g., temperature, pH, ...).

EXAMPLE 10. Let $\mathcal{E}$ be the enforcing set $\{(\varnothing, \{b^n\}) \mid n \text{ is even }\}$. We refer to each enforcer $(\varnothing, \{b^n\})$ as a 'brute' enforcer, because the empty set is included in every language. Thus if K satisfies $\mathcal{E}$, then K *must* contain the language $\{b^n \mid n \text{ is even }\}$. □

For an enforcing set $\mathcal{E}$, $\mathcal{L}(\mathcal{E}) = \{K \mid K \underline{\text{sat}}\ \mathcal{E}\}$ is the family of ($\mathcal{E}$-)satisfying languages. Similarly, the family of finite satisfying languages is defined by $\mathcal{L}_{\text{fin}}(\mathcal{E}) = \{K \mid K \text{ is finite and } K \in \mathcal{L}(\mathcal{E})\}$.

THEOREM 11. *Let $\mathcal{E}$ be an enforcing set. If $K_1, K_2, \ldots$ is an ascending sequence of languages with $K_i \underline{\text{sat}}\ \mathcal{E}$ for all $i \geq 1$, then $(\bigcup_{i \geq 1} K_i) \underline{\text{sat}}\ \mathcal{E}$.*

4.2. Computing is evolving through enforcing. Let $\mathcal{E}$ be an enforcing set. An enforcer $(X, Y) \in \mathcal{E}$ is *applicable to* a language K (*K-applicable*) if $X \subseteq K$. If (X, Y) is K-applicable, but $Y \cap K = \varnothing$, then (X, Y) is a *K-violator*.

Let K_0 be a language, $\mathcal{E}$ an enforcing set, and assume that it is not true that $K_0 \underline{\text{sat}}\ \mathcal{E}$, i.e., there are $(X, Y) \in \mathcal{E}$ with $X \subseteq K_0$ but $Y \cap K_0 = \varnothing$. Now add, for each of these K_0-violators (X, Y), at least one element of Y to K_0, and denote the (possibly infinite) superset of K_0 constructed non-deterministically in this way by K_1. Then clearly none of the K_0-violators is a K_1-violator, but some enforcers that were not applicable to K_0 may become now K_1-violators. If so, then repeat the construction described above, and so on.

This iterative 'repair procedure' is illustrated in Figure 1.

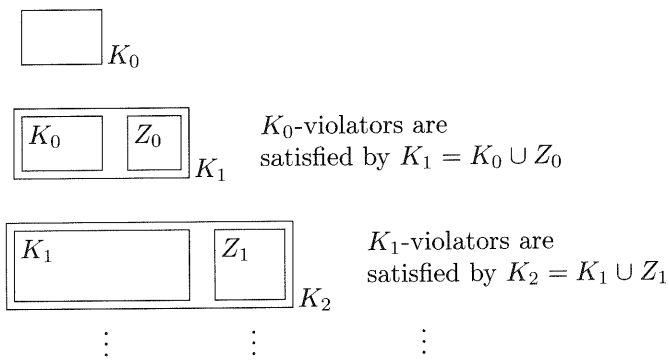

FIGURE 1.

The underlying idea of this 'evolving procedure' is formalized as follows.

DEFINITION 12. For an enforcing set $\mathcal{E}$ and languages K, K' with $K \subseteq K'$ we say that K' is an *$\mathcal{E}$-extension* of K, written $K \vdash_{\mathcal{E}} K'$, if $X \subseteq K$ implies $K' \cap Y \neq \varnothing$, for each $(X, Y) \in \mathcal{E}$. □

The $\mathcal{E}$-extension relation expresses the basic computation step induced by $\mathcal{E}$: a molecular system that has to satisfy $\mathcal{E}$ evolves according to $\vdash_{\mathcal{E}}$. It is also our basic notion for studying computations in forbidding-enforcing systems.

The following theorem says that the iterative repair procedure above yields the desired result.

THEOREM 13. *Let $\mathcal{E}$ be an enforcing set and let $K_1, K_2, \ldots$ be an infinite ascending sequence of languages. If $K_i \vdash_{\mathcal{E}} K_{i+1}$ for each $i \geq 1$, then $(\bigcup_{i\geq 1} K_i)$ sat $\mathcal{E}$.*

It is instructive to see that the previous result does not hold for finite ascending sequences : take for instance $\mathcal{E} = \{(\{a\}, \{b\}), (\{a, b\}, \{c\})\}$, and let $K_1 = \{a\}$ while $K_2 = \{a, b\}$. Then $K_1 \subseteq K_2$ and $K_1 \vdash_{\mathcal{E}} K_2$ whereas $K_1 \cup K_2$ does not satisfy $\mathcal{E}$.

This certainly agrees with our intuition : the molecular reactions go on all the time, providing that the needed components (molecules) are available. Thus such reactions may lead out of a finite language.

4.3. Finitary normal form. In this subsection we consider an important normal form for enforcing sets. We begin by distinguishing two finiteness properties of enforcing sets.

We say that two enforcing sets $\mathcal{E}_1$ and $\mathcal{E}_2$ are equivalent if $\mathcal{L}(\mathcal{E}_1) = \mathcal{L}(\mathcal{E}_2)$. For a finite language Z we define $\mathcal{E}(Z) = \{(X, Y) \in \mathcal{E} \mid X = Z\}$.

DEFINITION 14.
(1) An enforcing set $\mathcal{E}$ is *finitary* if, for each finite language Z, $\mathcal{E}(Z)$ is finite.
(2) An enforcing set $\mathcal{E}$ is *weakly finitary* if, for each finite language K_1 there exists a finite language K_2 such that $K_1 \vdash_{\mathcal{E}} K_2$. □

If an enforcing set $\mathcal{E}$ is finitary, it means that each premise set Z can have only a finite number of different consequence sets included in $\mathcal{E}$. This is a *syntactic* feature of $\mathcal{E}$. On the other hand the property of being weakly finitary is more of a *semantic* property – it says something about the effect that the enforcing specified by $\mathcal{E}$ has on finite languages. This effect is required to be 'continuous' : each finite language *can* always evolve according to $\mathcal{E}$ to a finite language. Thus one can start with a finite language and evolve it according to $\mathcal{E}$ in a smooth way without 'exploding in one step' into an infinite set.

The basic relationship between finitary and weakly finitary enforcing sets is given by the following result.

THEOREM 15.
(1) Every finitary enforcing set is weakly finitary.
(2) There exist weakly finitary enforcing sets that are not finitary.

Our next result says that languages K that satisfy finitary enforcing sets $\mathcal{E}$ play for such sets the role of the universe (Σ^+), meaning that $\mathcal{E}$ becomes weakly finitary for all finite languages within such K.

THEOREM 16. *Let $\mathcal{E}$ be a finitary enforcing set, and let K be a language such that K sat $\mathcal{E}$. For every finite language $L \subseteq K$, there exists a finite language $L' \subseteq K$ such that $L \vdash_{\mathcal{E}} L'$.*

Note that the above result does not hold if we require that $\mathcal{E}$ is weakly finitary rather than finitary. To see this consider the weakly finitary enforcing set $\mathcal{E} = \{(\varnothing, \{a, b^n\}) \mid n \geq 1\}$. Let $K = \{b^n \mid n \geq 1\}$, then obviously K sat $\mathcal{E}$. Now let

$L \subseteq K$ be finite and assume that $L \vdash_{\mathcal{E}} L'$ for a finite language L'. Let $m = \max\{n \mid b^n \in L'\}$ and consider the enforcer $E = (\varnothing, \{a, b^{m+1}\})$. Obviously L' sat E does not hold, contradicting $L \vdash_{\mathcal{E}} L'$. Hence L' cannot be finite.

The following theorem is one of the main results of the forbidding-enforcing theory.

THEOREM 17. *For every enforcing set there exists an equivalent finitary enforcing set.*

4.4. Finiteness. In our approach we allow infinite enforcing sets. Unfortunately one cannot restrict oneself to finite enforcing sets only.

LEMMA 18. *If $\mathcal{E}$ is a finite enforcing set, then there is an $n \geq 1$ such that, for each $K \in \mathcal{L}(\mathcal{E})$ and each L, if $K|_{\leq n} \subseteq L \subseteq K$, then $L \in \mathcal{L}(\mathcal{E})$.*

Consequently, if $\mathcal{E}$ is finite, then $\mathcal{L}(\mathcal{E})$ contains a finite language, viz., $K|_{\leq n}$. Thus, e.g., for the enforcing set $\mathcal{E}$ in Example 10 there is no equivalent finite enforcing set, since every language in $\mathcal{L}(\mathcal{E})$ contains the infinite set $\{b^n \mid n \text{ is even}\}$. This gives the following result.

LEMMA 19. *There are infinite enforcing sets for which there is no equivalent finite enforcing set.*

We define $\mathcal{E}|_{\leq n}$ to be $\{(X, Y) \in \mathcal{E} \mid |w| \leq n \text{ for all } w \in X \cup Y\}$. Analogous to the situation with forbidding sets, satisfaction of an enforcing set $\mathcal{E}$ by a language K is guaranteed by the satisfaction of certain specific finite parts of $\mathcal{E}$ by certain specific finite parts of K.

LEMMA 20. *K sat $\mathcal{E}$ if and only if $K|_{\leq i}$ sat $\mathcal{E}|_{\leq i}$ for every $i \geq 1$.*

Unlike for forbidding sets, finite languages are not particularly important for families of satisfying languages. This can be shown using the fact that we can construct any language K as a singleton family, by enforcing each of its elements. Similarly, we can enforce that every element of the complement of K be present whenever one of its elements is present. Formally, let K be a language over Σ, and let $\langle w_i \rangle_{i \in \mathbb{N}}$ be an arbitrary but fixed ordering of the words of K. Similarly, let $\langle v_i \rangle_{i \in \mathbb{N}}$ be an arbitrary but fixed ordering of the elements of $\Sigma^+ - K$. Now consider the enforcing set $\mathcal{E}_K = \{(\varnothing, \{w_1\})\} \cup \{(\{w_i\}, \{w_{i+1}\}) \mid i \geq 1\} \cup \{(\{v_i\}, \{v_1\}), (\{v_i\}, \{v_{i+1}\}) \mid i \geq 1\}$. Then $\mathcal{L}(\mathcal{E}_K) = \{K, \Sigma^+\}$, and clearly for every finite language K we have $\mathcal{L}_{\text{fin}}(\mathcal{E}_K) = \{K\}$, whereas for infinite K we have $\mathcal{L}_{\text{fin}}(\mathcal{E}_K) = \varnothing$. Hence for any two different infinite languages K and K' we have $\mathcal{L}_{\text{fin}}(\mathcal{E}_K) = \mathcal{L}_{\text{fin}}(\mathcal{E}_{K'}) = \varnothing$, whereas $\mathcal{L}(\mathcal{E}_K) \neq \mathcal{L}(\mathcal{E}_{K'})$.

5. Forbidding-enforcing systems

We will investigate now systems that combine forbidding and enforcing.

DEFINITION 21. A *forbidding-enforcing system* (fe system for short) is a construct $\Gamma = (\mathcal{F}, \mathcal{E})$, where $\mathcal{F}$ is a forbidding set and $\mathcal{E}$ is an enforcing set. □

The corresponding *forbidding-enforcing family* (fe family), denoted $\mathcal{L}(\mathcal{F}, \mathcal{E})$, consists of all languages that are both $\mathcal{F}$-consistent and $\mathcal{E}$-satisfying. Hence $\mathcal{L}(\mathcal{F}, \mathcal{E}) = \mathcal{L}(\mathcal{F}) \cap \mathcal{L}(\mathcal{E})$.

EXAMPLE 22. Let $\Sigma = \{a, b\}$ and let $\Gamma = (\mathcal{F}, \mathcal{E})$ be the fe system obtained by combining the forbidding set $\mathcal{F} = \{\{aa, bb\}, \{ab, ba\}\}$ from Example 2 and the enforcing set $\mathcal{E} = \{(\varnothing, \{b^n\}) \mid n \text{ is even}\}$ from Example 10. Then a language $K \subseteq \Sigma^+$ is in $\mathcal{L}(\mathcal{F}, \mathcal{E})$ if and only if $K = K' \cup \{b^n \mid n \text{ is even}\}$ where either $K' \subseteq ab^*$ or $K' \subseteq b^*a$. □

We carry over the 'finitary' qualification of enforcing sets to fe systems in an obvious way : an fe system $\Gamma = (\mathcal{F}, \mathcal{E})$ is *finitary* if and only if $\mathcal{E}$ is finitary.

5.1. Examples.

DNA molecules – their structure and processing. DNA molecules, partially single or double stranded, can be coded over a suitable alphabet of base pairs. For the matching base pairs we may use the symbols $\binom{A}{T}, \binom{T}{A}, \binom{C}{G}, \binom{G}{C}$. For the single stranded pieces we can use $\binom{A}{\cdot}, \binom{T}{\cdot}, \binom{C}{\cdot}, \binom{G}{\cdot}$ (upper strand) and $\binom{\cdot}{A}, \binom{\cdot}{T}, \binom{\cdot}{C}, \binom{\cdot}{G}$ (lower strand). For this example we consider languages over the alphabet Σ_{base} consisting of these twelve symbols.

We will give some natural requirements on a formal language representing the set of linear DNA molecules. These requirements can be formulated in our forbidding-enforcing framework.

First, of course, no proper molecule can have an unmatched base in the upper strand next to an unmatched base in the lower strand. This leads to forbidders $\{\binom{\sigma}{\cdot}\binom{\cdot}{\tau}\}$ and $\{\binom{\cdot}{\sigma}\binom{\tau}{\cdot}\}$ for each $\sigma, \tau \in \{A, T, C, G\}$.

Additionally, the molecule described by the string $\binom{A}{T}\binom{C}{G}\binom{C}{\cdot}\binom{C}{G}\binom{T}{\cdot}$ is also described by the inverted string $\binom{\cdot}{T}\binom{G}{C}\binom{\cdot}{C}\binom{G}{C}\binom{T}{A}$. We denote the operation of inversion, which is the composition of mirror image and replacing each symbol $\binom{\tau}{\sigma}$ in Σ_{base} by $\binom{\sigma}{\tau}$, by $\underline{\text{inv}}$. In this way we get an infinite number of enforcers, $(\{x\}, \{\underline{\text{inv}}\,(x)\})$ with x over the alphabet Σ_{base}.

The above set of forbidders and the set of enforcers together yield a fe system that admits only *correct* and *all* correct descriptions of linear DNA molecules.

The effect of cutting by restriction enzymes can easily be translated into enforcing rules. E.g., for the restriction enzyme TaqI (see, e.g., [**NEB**]) we have the enforcer $(\{x\binom{T}{A}\binom{C}{G}\binom{G}{C}\binom{A}{T}y\}, \{x\binom{T}{A}\binom{\cdot}{G}\binom{\cdot}{C}\})$ for each pair $x, y \in \Sigma^*_{base}$. Note that we have enforced only one of the two halves, the other follows by palindromicity and inversion.

Recombination is then modelled by reversing the rules. E.g., ligating two pieces with overhang GC (one resulting from cutting with TaqI and the other a sticky end produced by the restriction enzyme NarI (see, e.g., [**NEB**])) can be enforced by

$$\left(\{x\binom{T}{A}\binom{\cdot}{G}\binom{\cdot}{C}, \binom{C}{\cdot}\binom{G}{\cdot}\binom{C}{G}\binom{C}{G}y\}, \{x\binom{T}{A}\binom{C}{G}\binom{G}{C}\binom{C}{G}\binom{C}{G}y\}\right).$$

Splicing systems. In splicing systems [**HPP97**] the above operations are abstracted to the use of splicing rules. The effect of such a splicing rule (u_1, v_1, u_2, v_2), which says that two words $x_1u_1v_1y_1$ and $x_2u_2v_2y_2$ can be spliced to form the word $x_1u_1v_2y_2$ can be described by an enforcing set as follows : $\{(\{x_1u_1v_1y_1, x_2u_2v_2y_2\}, \{x_1u_1v_2y_2\}) \mid x_1, x_2, y_1, y_2 \in \Sigma^*\}$.

More attractively, and more directly, one may have the rules of the form $(\{x, y, r\}, \{w\})$ where r is the restriction enzyme (after all it is a molecule) and x and y are spliced into w according to r. This seems to be attractive because, as

various molecules are created during the evolution of such a system, new (restriction) enzymes may become available and so their effects will also be produced – in this way we can deal with dynamically changing sets of rules.

A splicing rule may be specified in its usual string representation $u_1\#v_1\$u_2\#v_2$, whereas an enzyme may be given by its amino acid encoding, hence by a word over an alphabet of 20 symbols.

Satisfiability problem. We explain now a representation of the satisfiability problem by fe systems using the following example. Let $\Psi = (\neg x_1 \vee x_3 \vee x_6) \wedge (\neg x_1 \vee x_2 \vee \neg x_6)$ be a Boolean formula in 3-conjunctive normal form (see, e.g., [**GJ79**]). It consists of two clauses, each of which has to be satisfied.

The two truth assignments to the variable x_i can be encoded as the strings $\mathtt{x}\underline{\mathrm{bin}}\,(i)\mathtt{f}$ for 'x_i is false' and $\mathtt{x}\underline{\mathrm{bin}}\,(i)\mathtt{t}$ for 'x_i is true', where $\underline{\mathrm{bin}}\,(i) \in \{\mathtt{0},\mathtt{1}\}^*$ is a suitable binary encoding of i. Any language coding a truth assignment must have exactly one of the assignments for each variable. This can be achieved by having the brute enforcers $(\varnothing, \{\mathtt{x}v\mathtt{f}, \mathtt{x}v\mathtt{t}\})$ for each $v \in \{\mathtt{0},\mathtt{1}\}^*$ (this constitutes the enforcing set $\mathcal{E}_U$), and by having the forbidder $\{\mathtt{x}v\mathtt{f}, \mathtt{x}v\mathtt{t}\}$ for each $v \in \{\mathtt{0},\mathtt{1}\}^*$ (this constitutes the forbidding set $\mathcal{F}_U$).

Besides these universally valid restrictions, the formula Ψ itself places additional restrictions on the truth assignment. For instance, the clause $\neg x_1 \vee x_3 \vee x_6$ demands to assign true either to $\neg x_1$, or to x_3, or to x_6. Equivalently, it demands not to assign false to all of $\neg x_1$, x_3, and x_6 at the same time, i.e., it forbids the words $\mathtt{x1t}$, $\mathtt{x11f}$, and $\mathtt{x110f}$ to occur at the same time. Hence, the first clause is represented by the forbidder $\{\mathtt{x1t}, \mathtt{x11f}, \mathtt{x110f}\}$ and the second clause by the forbidder $\{\mathtt{x1t}, \mathtt{x10f}, \mathtt{x110t}\}$.

In this way, representing each clause by a forbidder, we get the forbidding set $\mathcal{F}_\Psi$. Now the 'universal' forbidders and enforcers together with the forbidders defined by the formula Ψ yield the fe system $\Gamma_\Psi = (\mathcal{F}_U \cup \mathcal{F}_\Psi, \mathcal{E}_U)$. This Γ_Ψ provides a succinct representation for the satisfiability of Ψ: Ψ is satisfiable if and only if $\mathcal{L}(\Gamma_\Psi)$ is non-empty.

5.2. The structure of computation in forbidding-enforcing systems. We move now to consider the structure of computations in fe systems, and in particular we claim that for the systems in finitary normal form there is an elegant representation, in the form of a tree, of all the computations in a given fe system.

We use here the standard notion of a tree. The trees will be rooted, node-labelled, they may be infinite but always finitely branching. This means that each node has only a finite number of children (however, we do not assume that there is a common bound on the number of children for each node). The label of a node v in a tree τ is denoted by $\underline{\mathrm{lab}}_\tau(v)$. We call a path in a tree a *full* path if it starts at the root and either ends at a leaf or is infinite.

DEFINITION 23. Let $\Gamma = (\mathcal{F}, \mathcal{E})$ be an fe system, and let τ be a tree. Then τ is a Γ-*tree* if
(1) each node label is an element of $\mathcal{L}_{\mathrm{fin}}(\mathcal{F})$,
(2) if a node v_2 is a descendant of a node v_1, then $\underline{\mathrm{lab}}_\tau(v_1) \vdash_{\mathcal{E}} \underline{\mathrm{lab}}_\tau(v_2)$. □

Hence the influence of the forbidding set is expressed by the sort of node labels that are admitted, while the influence of the enforcing set is expressed through the condition on the sort of languages that can follow each other on a single path – this is illustrated in Figure 2.

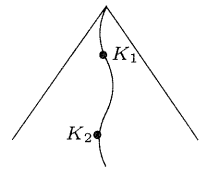

FIGURE 2. $K_1, K_2 \in \mathcal{L}_{\text{fin}}(\mathcal{F})$, $K_1 \subseteq K_2$, and $K_1 \vdash_{\mathcal{E}} K_2$

We now consider a Γ-tree where all the languages from $\mathcal{L}(\Gamma)$, finite and infinite, are represented.

DEFINITION 24. Let $\Gamma = (\mathcal{F}, \mathcal{E})$ be an fe system. A Γ-tree τ is a *complete* Γ-tree if
(1) if $K \subseteq \Sigma^+$ is finite and $K \in \mathcal{L}(\Gamma)$, then K is a node label of τ,
(2) if $K \subseteq \Sigma^+$ is infinite and $K \in \mathcal{L}(\Gamma)$, then there exists an infinite path π in τ such that $K = \bigcup_{v \in \pi} \underline{\text{lab}}_\tau(v)$. □

We know already that finitary fe systems constitute a normal form for forbidding-enforcing systems meaning that as far as the specifications of fe families are concerned one can restrict oneself to finitary fe systems. However, the real attractiveness of finitary fe systems stems from the following result.

THEOREM 25. *For each finitary fe system Γ there exists a complete Γ-tree.*

This means that every finitary fe system Γ can be 'completely' represented by a complete Γ-tree τ, that represents both *all languages* defined by Γ and *all computations* taking place within Γ :
(1) all finite languages in $\mathcal{L}(\Gamma)$ occur as node labels in τ,
(2) by taking for each infinite path the union of all languages along this path we get all infinite languages in $\mathcal{L}(\Gamma)$,
(3) by following all full paths in τ we get all evolving computations of Γ.

Since finitary fe systems form a normal form for fe systems, one can represent all languages defined by an arbitrary fe system Γ by a tree, viz., the Γ-tree of an equivalent finitary fe system. What will not carry over is the structure of computations in the original fe system.

THEOREM 26. *For each fe system Γ there exists a finitely branching tree τ with nodes labelled by finite languages such that for each language K,*
$K \in \mathcal{L}(\Gamma)$ iff there exists a full path π in τ such that $K = \bigcup_{v \in \pi} \underline{lab}_\tau(v)$.

5.3. The role of finite languages. As we have indicated already fe systems are novel also from formal language theoretic point of view in the sense that a single fe system defines a possibly infinite family of languages (rather than a single language – which is standard in formal language theory). However, there is another very important, and more 'semantical', difference with traditional formal language theory. In classical formal language theory finite languages are 'irrelevant' in the following sense : given any standard sort of grammar G (e.g. context-free grammar, context-sensitive grammar, ET0L system, ...) and a finite language F, the languages $L(G) - F$ and $L(G) \cup F$ can be defined by the same sort of grammar. Actually in all these grammars each finite language can be defined in a trivial (meaningless) way : every element of the finite language can be generated in one

step from the axiom of the grammar. The situation for fe systems is drastically different. An fe family is determined by certain finite parts of languages, as formalized by the following result.

THEOREM 27. *Let $\mathcal{K}$ be an fe family, and K a language. If, for all $n \geq 1$, there is an $L \in \mathcal{K}$ with $K|_{\leq n} = L|_{\leq n}$, then $K \in \mathcal{K}$.*

COROLLARY 28. *Let $\mathcal{K}$ be an fe family, and $\Delta \subseteq \Sigma$. If all finite languages over Δ are in $\mathcal{K}$, then all languages over Δ are in $\mathcal{K}$.*

COROLLARY 29. *Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be fe families. If $\{K|_{\leq n} \mid K \in \mathcal{K}_1\} = \{K|_{\leq n} \mid K \in \mathcal{K}_2\}$ for all $n \geq 1$, then $\mathcal{K}_1 = \mathcal{K}_2$.*

5.4. Some topology. Since investigation of computations in fe systems often leads to convergence problems of (evolving) sequences of languages, one comes in a natural way to some basic topological notions. The following theorem, a topological reformulation of Theorem 27, is a generalization of Theorem 3 (3) and Theorem 11, in the sense that we do not require an ascending sequence of languages, the union of which equals a certain language K, but rather a sequence of languages that converges to K.

THEOREM 30. *Let $\Gamma = (\mathcal{F}, \mathcal{E})$ be an fe system and let K be a language. If $K_1, K_2, \ldots$ is a sequence of languages with $\langle K_n \rangle_{n \in \mathbb{N}} \to K$ and $K_i \in \mathcal{L}(\mathcal{F}, \mathcal{E})$ for all $i \in \mathbb{N}$, then $K \in \mathcal{L}(\mathcal{F}, \mathcal{E})$.*

This is exactly the definition of closed sets (in topological sense) : thus the language sets $\mathcal{L}(\mathcal{F}, \mathcal{E})$ are closed sets. In fact, it can be shown that the sets of languages for which a tree as in Theorem 26 exists, are exactly the closed sets (of languages).

6. Discussion

In this paper we have introduced a model of computation which we believe is novel from both molecular systems and computation theory point of view. We believe that forbidding-enforcing is an interesting paradigm to be investigated. We see (at least) three lines of research continuing this paper :

(1) Investigating the use of fe systems for the description of various types of computing, both dry and wet. Examples given in Section 5.1 form only a beginning of a step in this direction.

(2) Developing the theory of fe systems. In particular we plan here to explore Theorem 25, and to investigate basic decision problems and complexity issues. There are also many combinatorial and topological questions that, in view of the current paper, should be pursued.

(3) There is nothing in the basic idea of forbidding-enforcing systems that restricts them to strings only. It is well known that graphs are useful for describing the basic structure of molecules. Moreover, using forbidden subgraphs to specify classes of graphs (e.g., planar graphs) is quite well understood. We plan to investigate fe systems over graphs.

References

[Amo97] Martyn Amos. *DNA Computation.* PhD thesis, Department of Computer Science, University of Warwick, UK, September 1997.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* Bell Telephone Laboratories, 1979.

[HPP97] Thomas Head, Gheorghe Păun, and Dennis Pixton. Language theory and molecular genetics : Generative mechanisms suggested by DNA recombination. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 2, chapter 7, pages 295–360. Springer-Verlag, 1997.

[NEB] New England BioLabs Catalog 1998/1999.

[Win98] Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, Pasadena, California, May 19 1998.

(A. Ehrenfeucht) Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA

(H. J. Hoogeboom) LIACS, Universiteit Leiden, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

E-mail address: hoogeboo@wi.leidenuniv.nl

(G. Rozenberg) LIACS, Universiteit Leiden, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands, and, Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA

E-mail address: rozenber@wi.leidenuniv.nl

(N. van Vugt) LIACS, Universiteit Leiden, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

E-mail address: nvvugt@wi.leidenuniv.nl