Viral Gene Compression: Complexity and Verification*

Mark Daley[1,2] and Ian McQuillan[2]

[1] University of Saskatchewan, Saskatoon, Saskatchewan, Canada
daley@cs.usask.ca
[2] University of Western Ontario, London Ontario, Canada
{imcquill, daley}@csd.uwo.ca

Abstract. The smallest known biological organisms are, by far, the viruses. One of the unique adaptations that many viruses have aquired is the compression of the genes in their genomes. In this paper we study a formalized model of gene compression in viruses. Specifically, we define a set of constraints that describe viral gene compression strategies and investigate the properties of these constraints from the point of view of genomes as languages. We pay special attention to the finite case (representing real viral genomes) and describe a metric for measuring the level of compression in a real viral genome. An efficient algorithm for establishing this metric is given along with applications to real genomes including automated classification of viruses and prediction of horizontal gene transfer between host and virus.

1 Introduction

In contrast to the lengthy, often redundant, genomes of higher organisms, the genomes of viruses are extremely efficient in the encoding of their genes. Where mammalian genomes, for example, possess lengthy introns which code for no genes at all, any given segment of a viral genome may be a coding region for several genes. In addition to prefix and suffix overlap of viral genes, some genes may also be encoded in a retrograde fashion (that is, the gene would be read in a direction opposite to other genes). These systems provide evidence that viruses have evolved a special type of information compression technique. Studying this natural compression system in a rigorous setting could yield insight into the structure of viral genomes and may contribute to a basis for classifying such structures.

In this paper, we will specifically consider the types of compression seen in two small double-stranded DNA virus families, Papillomavirus and Polyomavirus, and single-stranded RNA viruses from the Bornavirus, Coronavirus and, to a lesser extent, the Filovirus and Retrovirus families.

* This research was funded in part by institutional grants of the University of Saskatchewan (M. Daley), the University of Western Ontario (M. Daley) and the Natural Science and Engineering Research Council of Canada (M. Daley and H. Jürgensen).

The importance of this genetic compression becomes obvious when considering the structure of viruses. Viruses generally consist of two principal components: a protein capsid, and genetic material inside the capsid. The capsid serves as protection for the genetic material and also as a mechanism for inserting the genetic material into a host cell. The genetic material may consist of single- or double-stranded DNA or RNA and, in some rare cases, a mixture of the former possibly also including proteins.

The need for compression stems from the fact that the size of the capsid limits the amount of room for genetic information inside the virus. In the case of Polyomaviruses, the genome is constrained to be approximately 5kbp (5,000 basepairs) of DNA (compare to the human genome of size 3,150,000 kb), yet still manages to encode 6 distinct genes.

We exposit here a formal model of the viral compression techniques in terms of constraints on languages. For example, we would say that a language satisfies the "viral overlapping compression" property if some prefix of some word in the language is also a suffix of some other word in the language. We can likewise define constraints for other viral compression techniques, including retrograde encodings. We will focus here on deterministic modeling of the gene-level mechanics, in contrast to the probabilistic analysis of [8], which addresses gene compression from the point of view of evolutionary pressures and constraints on entire genomes.

The organization of the paper is as follows: In section 2 we consider basic notation and prerequisites. In section 3 we define formal versions of the basic viral compression techniques and investigate relationships and dependencies between them. We consider also the question of for which families of languages it is possible to decide these properties. Section 4 focuses on the finitary case of the problem as this is the most interesting from the point of view of applied viral genetics. We present efficient algorithms to decide each of the properties for real viral genomes. Section 5 contains our conclusions and a discussion of practical applications.

2 Notation and Prerequisites

For a general introduction to virology, we refer the reader to [3] and [10]; for formal language theory preliminaries, we refer to [9]. Let Σ be a finite alphabet. We denote, by Σ^* and Σ^+, the sets of words and non-empty words, respectively, over Σ and the empty word by λ. A language L is any subset of Σ^*. For a word $w \in \Sigma^*$, we denote the length of w by $|w|$ and the reversal of w by w^R. Let $\mathbb{N}$ be the set of positive integers. Furthermore, for $k \in \mathbb{N}$, define $\Sigma^{\geq k} = \{w \in \Sigma^* \mid |w| \geq k\}$.

A *full trio* is a language family closed under homomorphism, inverse homomorphism and intersection with regular sets. A full trio is also referred to as a cone. It is known that every full trio is closed under arbitrary a-transductions[1] and hence arbitrary gsm mappings. We refer to [1, 4] for the theory of AFL's.

[1] An a-transducer is also referred to as a rational transducer.

For a binary relation $\varrho \subseteq \Sigma^* \times \Sigma^*$ and a language $L \subseteq \Sigma^*$, we define

$$\begin{aligned}
&\varrho(s) = \{t \in \Sigma^* \mid (s,t) \in \varrho\},\\
&\varrho(L) = \{t \in \Sigma^* \mid s \in L, (s,t) \in \varrho\},\\
&\varrho^R = \{(s,t^R) \mid (s,t), \in \varrho\},\\
&\varrho^{-1} = \{(t,s) \mid (s,t) \in \varrho\},\\
&\varrho^{-R} = (\varrho^{-1})^R.
\end{aligned}$$

We will consider the following well-known relations. Let $w, v \in \Sigma^*$.

1. prefix order: $w \leq_{\mathrm{p}} v$ (or $(w,v) \in \leq_{\mathrm{p}}$) if and only if $v = wx$ for some $x \in \Sigma^*$.
2. suffix order: $w \leq_{\mathrm{s}} v$ (or $(w,v) \in \leq_{\mathrm{s}}$) if and only if $v = xw$ for some $x \in \Sigma^*$.
3. infix order: $w \leq_{\mathrm{i}} v$ (or $(w,v) \in \leq_{\mathrm{i}}$) if and only if $v = xwy$ for some $x, y \in \Sigma^*$.

Also, for each of the relations above, we prepend the word "proper", which will be denoted by $<_{\mathrm{p}}, <_{\mathrm{s}}, <_{\mathrm{i}}$ where we enforce that $x, y \in \Sigma^+$ above.

For example, $\leq_{\mathrm{p}} (L) = \{w \in \Sigma^* \mid v \leq_{\mathrm{p}} w, v \in L\}$ and $\leq_{\mathrm{i}}^{-R} (L) = \{w \in \Sigma^* \mid xw^R y \in L, x, y \in \Sigma^*\}$.

3 Viral Properties

Before formally stating the definitions of the viral properties, we will define the following sets which will be used for the conditions.

Let $L \subseteq \Sigma^*$ be a language, and let $n \in \mathbb{N}$ such that $1 \leq n \leq 6$ and let $k \in \mathbb{N}$. Then we define the following sets:

$$\begin{aligned}
&U(1,L,k) = \{w \in \Sigma^* \mid \exists u \in \Sigma^{\geq k}, x \in \Sigma^+, v \in \Sigma^*, xu \in L, w = uv\},\\
&U(2,L,k) = \{w \in \Sigma^* \mid \exists v \in \Sigma^{\geq k}, y \in \Sigma^+, u \in \Sigma^*, vy \in L, w = uv\},\\
&U(3,L,k) = U(1,L,k) \cup U(2,L,k),\\
&U(4,L,k) = \{w \in \Sigma^* \mid \exists u, v \in \Sigma^{\geq k}, x, y \in \Sigma^+, xu \in L \wedge vy \in L, w = uv\},\\
&U(5,L,k) = \{w \in \Sigma^{\geq k} \mid \exists u, v \in \Sigma^*, uw^R v \in L\},\\
&U(6,L,k) = \{w \in \Sigma^{\geq k} \mid \exists u, v \in \Sigma^*, uw^R v \in L^+\}.
\end{aligned}$$

Furthermore, for each i, $1 \leq i \leq 6$, $k \in \mathbb{N}$ and $L \subseteq \Sigma^*$, let $Z(i,L,k) = U(i,L,k) \cap L$.

So, for example, $Z(i,L,k)$ consists of all words $w \in L$ such that there exists a word u of length at least k, a non-empty word x and a word v whereby xu is in L and $w = uv$ which is also in L.

We now define the properties that we will study.

Definition 1. *Let $L \subseteq \Sigma^*$, let n satisfy $1 \leq n \leq 6$ and let $k, l \in \mathbb{N}$. We say that L satisfies condition $W(n,k,l)$ if $|Z(n,L,k)| \geq l$.*

We also call condition $W(1,k,l)$ the *l-weak k-prefix overlapping property*, condition $W(2,k,l)$ the *l-weak k-suffix overlapping property*, condition $W(3,k,l)$

the *l-weak k-overlapping property*, condition $W(4,k,l)$ the *l-weak k-double-sided overlapping property*, condition $W(5,k,l)$ the *l-weak k-retrograde overlapping property* and condition $W(6,k,l)$ the *l-weak k-concatenated retrograde overlapping property.*

For example, a language L satisfies $W(1,k,l)$ if and only if there exists l distinct words $w \in L$ whereby $w = uv$ for some $u, v, x \in \Sigma^*$, with u of length at least k,x non-empty and $xu \in L$.

We also define a strong version of the properties above.

Definition 2. *Let $L \subseteq \Sigma^*$, let n satisfy $1 \leq n \leq 6$ and let $k \in \mathbb{N}$. We say that L satisfies condition $V(n,k)$ if $L \subseteq U(n,L,k)$. Equivalently, L satisfies condition $V(n,k)$ if and only if $L = Z(n,L,k)$.*

We also refer to each of these properties by replacing the prefix "l-weak" of each condition above with "strong".[2]

We now consider the relationships of these properties to each other. The following is immediate from the definitions.

Lemma 1. *Let i, j satisfy $1 \leq i, j \leq 6$, $k, l \in \mathbb{N}$ and let $L \subseteq \Sigma^*$. Then the following are true:*

1. *If $U(i,L,k) \subseteq U(j,L,k)$ then both L satisfies $V(i,k)$ implies L satisfies $V(j,k)$ and L satisfies $W(i,k,l)$ implies L satisfies $W(j,k,l)$.*
2. *If $|L \cap \Sigma^{\geq k}| \geq l$, then L satisfies $V(i,k)$ implies L satisfies $W(i,k,l)$.*
3. *If $L \cap \Sigma^{\leq k} \neq \emptyset$, then L does not satisfy $V(i,k)$.*

Also, we note the following:

Lemma 2. *Let $L \subseteq \Sigma^*, k \in \mathbb{N}$. Then the following are true:*

1. *$U(4,l,k) \subseteq U(i,l,k)$, for each $i \in \{1,2,3\}$,*
2. *$U(5,l,k) \subseteq U(6,l,k)$,*
3. *$Z(1,L,k)^R = Z(2,L^R,k)$ and thus $|Z(1,L,k)| = |Z(2,L^R,k)|$.*

Proof. The first three statements are straightforward. For the fourth statement, let $z \in Z(1,L,k)^R$. Thus, $z^R = uv \in L, xu \in L, u \in \Sigma^{\geq k}, x \in \Sigma^+, v \in \Sigma^*$. Then $z = v^R u^R \in L^R, u^R x^R \in L^R$ and $z \in Z(2,L^R,k)$. Conversely, let $z \in Z(2,L^R,k)$. Thus, $z = uv, vy \in L^R, v \in \Sigma^{\geq k}, y \in \Sigma^+, u \in \Sigma^*$. Then $v^R u^R, y^R v^R \in L, v^R u^R \in Z(1,L,k)$ and $z = uv \in Z(1,L,k)^R$. □

Combining Lemma 1, 2, we obtain:

Proposition 1. *Let $L \subseteq \Sigma^*, k, l \in \mathbb{N}$. Then the following statements are true:*

1. *L satisfies $W(1,k,l)$ or $W(2,k,l)$ implies L satisfies $W(3,k,l)$,*
2. *L satisfies $V(1,k)$ or $V(2,k)$ implies L satisfies $V(3,k)$,*

[2] While prefix/suffix overlap compression is very common in viruses, it is not often the case that *every* gene will have some overlap; hence the motivation to study "weakened" versions of these operations.

3. *L satisfies $W(4,k,l)$ implies L satisfies $W(1,k,l)$, $W(2,k,l)$ and $W(3,k,l)$,*
4. *L satisfies $V(4,k)$ implies L satisfies $V(1,k)$, $V(2,k)$ and $V(3,k)$,*
5. *L satisfies $W(5,k,l)$ implies L satisfies $W(6,k,l)$,*
6. *L satisfies $V(5,k)$ implies L satisfies $V(6,k)$.*
7. *L (respectively L^R) satisfies $W(1,k,l)$ iff L^R (resp. L) satisfies $W(2,k,l)$,*
8. *L (respectively L^R) satisfies $V(1,k)$ iff L^R (resp. L) satisfies $V(2,k)$.*

We see, however that if $L_1 = \{abc, aa\}$ and $L_2 = \{abc, cc\}$, then $Z(1, L_1, 1) = \{aa, abc\}$, $Z_2(2, L_1, 1) = \{aa\}$, $Z(1, L_2, 1) = \{cc\}$ and also $Z(2, L_2, 1) = \{abc, cc\}$. So, in general, there are languages satisfying $W(1,k,l)$ (respectively $V(1,k)$) but not $W(2,k,l)$ (respectively $V(2,k)$) and there are languages satisfying $W(2,k,l)$ (respectively $V(2,k)$) but not $W(1,k,l)$ (respectively $V(1,k)$). We note also that, since $Z(3, L_1, 1) = Z(1, L_1, 1) \cup Z(2, L_1, 1)$ and $Z(3, L_2, 1) = Z(1, L_2, 1) \cup Z(2, L_2, 1)$, there are languages satisfying $W(3,k,l)$ (respectively $V(3,k)$) but not $W(1,k,l)$ (respectively $(V(1,k))$ and there are languages satisfying $W(3,k,l)$ (respectively $V(3,k)$) but not $W(2,k,l)$ (respectively $V(2,k)$). Additionally, $Z(4, L_1, 1) = \{aa\}$ and $Z(4, L_2, 1) = \{cc\}$. Thus, in general, there are languages satisfying $W(1,k,l)$ (respectively $V(1,k)$) but not satisfying $W(4,k,l)$ (respectively $V(4,k)$), there are languages satisfying $W(2,k,l)$ (respectively $V(2,k)$) but not satisfying $W(4,k,l)$ (respectively $V(4,k)$) and there are languages satisfying $W(3,k,l)$ (respectively $V(3,k)$) but not satisfying $W(4,k,l)$ (respectively $V(4,k)$. Further, let $L_3 = \{a, b, c, abc\}$. Then $Z(5, L_3, 1) = \{a, b, c\}$ but $Z(6, L_3, 1) = L_3$ and so, in general, there are languages satisfying $W(6,k,l)$ (respectively $V(6,k,l)$) but not $W(5,k,l)$ (respectively $V(6,k,l)$).

We also define the following sets which we will use for a characterization.

$$\begin{aligned}
C(1,L,k) &= \leq_{\mathrm{p}} (<_{\mathrm{s}}^{-1} (L) \cap \Sigma^{\geq k}),\\
C(2,L,k) &= \leq_{\mathrm{s}} (<_{\mathrm{p}}^{-1} (L) \cap \Sigma^{\geq k}),\\
C(3,L,k) &= C(1,L,k) \cup C(2,L,k),\\
C(4,L,k) &= (<_{\mathrm{s}}^{-1} (L) \cap \Sigma^{\geq k}) \cdot (<_{\mathrm{p}}^{-1} (L) \cap \Sigma^{\geq k}),\\
C(5,L,k) &= \leq_{\mathrm{i}}^{-R} (L) \cap \Sigma^{\geq k},\\
C(6,L,k) &= \leq_{\mathrm{i}}^{-R} (L^{+}) \cap \Sigma^{\geq k}.
\end{aligned}$$

Proposition 2. *Let i satisfy $1 \leq i \leq 6$, let $k \in \mathbb{N}$ and let $L \subseteq \Sigma^*$. Then $U(i,L,k) = C(i,L,k)$.*

Proof. Let $i = 1$. "$\subseteq$" Let $w \in U(1,L,k)$. Thus, there exists $u \in \Sigma^{\geq k}, v \in \Sigma^*, x \in \Sigma^+, xu \in L, w = uv$. Therefore, $u \in <_{\mathrm{s}}^{-1} (L) \cap \Sigma^{\geq k}$ and $w \in \leq_{\mathrm{p}} (<_{\mathrm{s}}^{-1} (L) \cap \Sigma^{\geq k})$.

"$\supseteq$" Let $w \in \leq_{\mathrm{p}} (<_{\mathrm{s}}^{-1} (L) \cap \Sigma^{\geq k})$. Thus, there exists $u, v \in \Sigma^*$ such that $w = uv$ with $u \in <_{\mathrm{s}}^{-1} (L) \cap \Sigma^{\geq k}$. Hence, there exists $x \in \Sigma^+$ such that $xu \in L$.

Let $i = 2$. "$\subseteq$" Let $w \in U(2,L,k)$. Thus, there exists $v \in \Sigma^{\geq k}, u \in \Sigma^*, y \in \Sigma^+, vy \in L, w = uv$. Therefore, $v \in <_{\mathrm{p}}^{-1} (L) \cap \Sigma^{\geq k}$ and $w \in \leq_{\mathrm{s}} (<_{\mathrm{p}}^{-1} (L) \cap \Sigma^{\geq k})$.

"$\supseteq$" Let $w \in \leq_{\mathrm{s}} (<_{\mathrm{p}}^{-1} (L) \cap \Sigma^{\geq k})$. Thus, there exists $u, v \in \Sigma^*$ such that $w = uv$ with $v \in <_{\mathrm{p}}^{-1} (L) \cap \Sigma^{\geq k}$. Hence, there exists $x \in \Sigma^+$ such that $vx \in L$.

Let $i = 3$. Immediate from case 1, 2.

Let $i = 4$. "$\subseteq$" Let $w \in U(4, L, k)$. Thus, there exists $u, v \in \Sigma^{\geq k}, x, y \in \Sigma^+, w = uv, (xu \in L \wedge vy \in L)$. Therefore, $u \in <_{\mathrm{s}}^{-1} (L) \cap \Sigma^{\geq k}, v \in <_{\mathrm{p}}^{-1} (L) \cap \Sigma^{\geq k}$ and $w \in (<_{\mathrm{s}}^{-1} (L) \cap \Sigma^{\geq k})(<_{\mathrm{p}}^{-1} (L) \cap \Sigma^{\geq k})$.

"$\supseteq$" Let $w \in (<_{\mathrm{s}}^{-1})(L) \cap \Sigma^{\geq k})(<_{\mathrm{p}}^{-1} (L) \cap \Sigma^{\geq k})$. Thus, there exists $u \in <_{\mathrm{s}}^{-1} (L) \cap \Sigma^{\geq k}, v \in <_{\mathrm{p}}^{-1} (L) \cap \Sigma^{\geq k}$ with $w = uv$. Hence, there exists $x, y \in \Sigma^+$ such that $xu \in L$ and $vy \in L$.

Let $i = 5$. "$\subseteq$" Let $w \in U(5, L, k)$. Thus, there exists $u, v \in \Sigma^*$ with $uw^Rv \in L$ and $w \in \Sigma^{\geq k}$. Then $w \in \leq_{\mathrm{i}}^{-R} (L) \cap \Sigma^{\geq k}$.

"$\supseteq$" Let $w \in \leq_{\mathrm{i}}^{-R} (L) \cap \Sigma^{\geq k}$. Then there exists u, v with $uw^Rv \in L$ and $w \in \Sigma^{\geq k}$.

Let $i = 6$. "$\subseteq$" Let $w \in U(6, L, k)$. Thus, there exists $u, v \in \Sigma^*$ with $uw^Rv \in L^+, w \in \Sigma^{\geq k}$. Then $w \in \leq_{\mathrm{i}}^{-R} (L^+) \cap \Sigma^{\geq k}$.

"$\supseteq$" Let $w \in \leq_{\mathrm{i}}^{-R} (L^+) \cap \Sigma^{\geq k}$. Then there exists u, v with $uw^Rv \in L^+, w \in \Sigma^{\geq k}$. $\square$

This leads naturally to some decision problems. One would like to provide algorithms to test whether languages (or genomes) satisfy these properties. Namely, can we decide whether a given language satisfies one of the properties, depending on the language family that the given language is in? For each weak condition, this amounts to deciding whether $|Z(i, L, k)| \geq l$ and for each strong condition, it amounts to deciding whether $Z(i, L, k) = L$.

Proposition 3. *Let $\mathcal{L}_1, \mathcal{L}_2$ be language families effectively closed under intersection and the full trio operations with $\mathcal{L}_1$ being effectively semilinear and $\mathcal{L}_2$ having a decidable equality problem. Then the following are true:*

1. *For each $k, l \in \mathbb{N}$ and $i, 1 \leq i \leq 4$, it is decidable whether $L \in \mathcal{L}_1$ satisfies $W(i, k, l)$ and it is decidable whether $L \in \mathcal{L}_2$ satisfies $V(i, k)$.*
2. *If $\mathcal{L}_1, \mathcal{L}_2$ are also effectively closed under reversal, then it is decidable whether $L \in \mathcal{L}_1$ satisfies $W(5, k, l)$ and it is decidable whether $L \in \mathcal{L}_2$ satisfies $V(5, k)$.*
3. *If $\mathcal{L}_1, \mathcal{L}_2$ are also effectively closed under reversal and $+$, then it is decidable whether $L \in \mathcal{L}_1$ satisfies $W(6, k, l)$ and it is decidable whether $L \in \mathcal{L}_2$ satisfies $V(6, k)$.*

Proof. It is easy to construct a-transducers which output $\leq_{\mathrm{p}}^{-1} (L), \leq_{\mathrm{s}}^{-1} (L), <_{\mathrm{p}}^{-1} (L), <_{\mathrm{s}}^{-1} (L), \leq_{\mathrm{p}} (L), \leq_{\mathrm{s}} (L)$ for each L in $\mathcal{L}_1$ or $\mathcal{L}_2$. Also, every intersection-closed full trio is closed under union and concatenation since $L_1\$\Sigma^* \cap \Sigma^*\$L_2$ is in $\mathcal{L}_1$ and $\mathcal{L}_2$, there is an a-transducer which outputs $L_1 \cup L_2$ and there is a homomorphism which outputs L_1L_2. Thus, $Z(1, L, k)$, $Z(2, L, k)$, $Z(3, L, k)$, $Z(4, L, k)$ are in $\mathcal{L}_1$ and $\mathcal{L}_2$. Additionally, if $\mathcal{L}_1, \mathcal{L}_2$ are closed under reversal, then $Z(5, L, k)$ is in $\mathcal{L}_1$ and $\mathcal{L}_2$ and if $\mathcal{L}_1, \mathcal{L}_2$ are closed under reversal and $+$, then $Z(6, L, k)$ is in $\mathcal{L}_1$ and $\mathcal{L}_2$. Since $\mathcal{L}_1$ is effectively semilinear, we can decide if $L \in \mathcal{L}_1$ is infinite [5] and if it is not, then we can effectively find the length of the longest word in L. Then, we can test membership of every word of length less than or equal to that length to determine whether $|Z(i, L, k)| \geq l$

(emptiness is always decidable for semilinear sets, and since $\mathcal{L}_1$ is closed under intersection with regular languages, we can decide whether $w \in L$ by testing whether $L \cap \{w\} \neq \emptyset$). Also, by the decidability of equality for $\mathcal{L}_2$, the proposition follows. □

We denote by **NCM** the family of languages defined by one-way nondeterministic, reversal-bounded multicounter machines. It is known that **NCM** is an intersection and reversal closed full trio effectively closed under semilinearity [7]. Also, it is known that the family of regular languages is closed under all of the operations above and has a decidable equality problem.

Corollary 1. *For each $L \in$ **NCM**, each i, $1 \leq i \leq 5$ and each $k, l \in \mathbb{N}$, it is decidable whether L satisfies $W(i,k,l)$. In addition, for each $L \in$ **REG**, each i, $1 \leq i \leq 6$ and each $k, l \in \mathbb{N}$, it is decidable whether L satisfies $W(i,k,l)$ and $V(i,k)$.*

4 Computational Verification of Viral Properties

Ideally, one would like to apply the formal definitions given here to real viral genomes as a method for classifying viruses based on gene compression. In this section we will consider fast algorithms to do exactly this, and their complexity. Since all real viral genomes are finite, we will restrict ourselves to dealing with finite input languages here. We will describe algorithms which will verify each of the viral properties for a given input viral genome. A viral genome is a finite language in which the words are the genes of the virus.

For a finite language $L \subseteq \Sigma^+$, we let s_L be the sum of the lengths of every word of L (the length of the genome).

We recall a well-known and important result from [2]. A partial deterministic finite automaton is a deterministic finite automaton in which each state need not have a transition on every letter. The smallest partial DFA for a given regular language is the partial DFA that recognizes the language and has the smallest number of states. In [2], it is demonstrated that, for each word $w \in \Sigma^*$, the smallest partial DFA accepting $\leq_{\mathrm{s}}^{-1}(w)$ is linear in the length of w. Precisely, it has at most $2|w| - 1$ states and $3|w| - 4$ transitions. Moreover, it is shown that the smallest partial DFA accepting $\leq_{\mathrm{i}}^{-1}(w)$ is linear in the length of w. That is, if $|w| > 2$, then it has at most $2|w| - 2$ states and at most $3|w| - 4$ transitions. In addition, they show that for any w over a fixed finite alphabet Σ, both the smallest partial DFA accepting $\leq_{\mathrm{s}}^{-1}(L)$ and the smallest DFA accepting $\leq_{\mathrm{i}}^{-1}(L)$ can be built in time linear in the length of w.

Now, let $L = \{w_1, \ldots, w_m\} \subseteq \Sigma^+$. For our algorithms, we construct a method which we call *suffix_dfa*(L) which returns a DFA accepting $\leq_{\mathrm{s}}^{-1}(L)$. Let $w = w_1\#w_2\# \cdots \#w_m\#$. Then $\leq_{\mathrm{s}}^{-1}(w) = (\leq_{\mathrm{s}}^{-1}(w_m\#)) \cup (\leq_{\mathrm{s}}^{-1}(w_{m-1}\#)w_m\#) \cup \ldots \cup (\leq_{\mathrm{s}}^{-1}(w_1\#)w_2\#w_3\# \cdots w_m\#)$. Let $M = (Q, \Sigma \cup \{\#\}, q_0, F, \delta)$ be the smallest partial DFA accepting $\leq_{\mathrm{s}}^{-1}(w)$. Thus, it is clear that for every $x \in \Sigma^*$, $x \in \leq_{\mathrm{s}}^{-1}(L)$ if and only if $x\#v \in \leq_{\mathrm{s}}^{-1}(w) = L(M)$, where $v \in (\Sigma \cup \{\#\})^*$. Moreover, since M is partial and the smallest DFA, for each $y \in (\Sigma \cup \{\#\})^*$,

$\delta(q_0, y)$ is defined if and only if $yu \in L(M)$ for some $u \in (\Sigma \cup \{\#\})^*$. Thus, for each $x \in \Sigma^*$, $x \in \leq_{\mathrm{s}}^{-1} (L)$ if and only if $\delta(q_0, x\#)$ is defined. Hence, we transform M into a new DFA M' by making the new final state set F' to be the set of all states $q \in Q$ such that $\delta(q, \#)$ is defined, and by removing all transitions of the form $\delta(q, \#) = p$ for $p, q \in Q$. Let $w \in L(M')$. Then $w \in \Sigma^*$ since there are no transitions on $\#$ and necessarily $w\#$ is defined in M. Thus, $w\#v \in L(M)$ for some $v \in (\Sigma \cup \{\#\})^*$. Thus, $w \in \leq_{\mathrm{s}}^{-1} (L)$. Conversely, let $w \in \leq_{\mathrm{s}}^{-1} (L)$. Then $w\#v \in L(M)$ for some $v \in (\Sigma \cup \{\#\})^*$ and so $w \in L(M')$. Hence we see that $L(M') = \leq_{\mathrm{s}}^{-1} (L)$ and M' can be constructed in linear time from M which is linear in $|w|$ which is linear in s_L. We note that *suffix_dfa*$(\Sigma^{-1}L) = <_{\mathrm{s}}^{-1} (L)$. Further, for a DFA $M = (Q, \Sigma, q_0, F, \delta)$ over Σ and $w \in \Sigma^*$, define $S_{M,k}(w) = \{q \in Q \mid \delta(q_0, w_1) = q, w_1 \leq_{\mathrm{p}} w, |w_1| \geq k\}$. For each algorithm in this section, we assume that we have some encoding of L as input, whereby there is only one copy of each word given.

Algorithm 1 input: $k \in \mathbb{N}, L \subseteq \Sigma^+$, Σ fixed, L finite, returns: largest l_1, l_2, l_3 such that L satisfies $W(1, k, l_1), W(2, k, l_2)$ and $W(3, k, l_2)$

```
1: Let $l_1, l_2, l_3 := 0, v_1, v_2 := false$, if $k \geq s_L$, return.
2: Let $M = (Q_1, \Sigma \cup \{\#\}, q_0, F_1, \delta_1) :=$ suffix_dfa$(\Sigma^{-1}L)$,
3: Let $M^R = (Q_2, \Sigma \cup \{\#\}, p_0, F_2, \delta_2) :=$ suffix_dfa$((L\Sigma^{-1})^R)$
4: for all $w \in L$ do
5:    if $S_{M,k}(w) \cap F_1 \neq \emptyset$ then
6:       $v_1 := true, l_1 := l_1 + 1$
7:    end if
8:    if $S_{M^R,k}(w^R) \cap F_2 \neq \emptyset$ then
9:       $v_2 := true, l_2 := l_2 + 1$,
10:   end if
11:   if either $v_1$ or $v_2$ is true, then
12:      $l_3 := l_3 + 1, v_1 := false, v_2 := false$.
13:   end if
14: end for
```

We have discussed above how to perform the method *suffix_dfa*. It is easy to pass in the reversal of a language to *suffix_dfa*, in time linear in s_L. Then, in line 5 of Algorithm 1, we can check to see if the intersection is empty by keeping a counter starting at k and running w through the transition function of M, decreasing the counter at each step. Then, when the counter reaches 0, we test every state we hit on input w to see whether it is a final state. If it is, we increase l_1 and set v_1 indicating that $w \in Z(1, L, k)$. Also, in line 8, we are testing whether $w^R \in Z(1, L^R, k)$. Indeed, by Lemma 2(3), $w^R \in Z(1, L^R, k)$ if and only if $w \in Z(2, L, k)$. Thus, if this is true, we increase l_2 and set v_2 to true. In addition, $w \in Z(3, L, k)$ if and only if $w \in Z(1, L, k) \cup Z(2, L, k)$ and so we increase l_3 if and only if either v_1 or v_2 is true, and we reset each to false. In this way, when the method completes, l_1, l_2 and l_3 will be the maximum such that L satisfies $W(1, k, l_1), W(2, k, l_2)$ and $W(3, k, l_3)$, respectively. Furthermore, this method runs in time $O(s_L)$ time.

For the fourth property, our algorithm requires only a small modification. For a word w, let $w(i)$ be the i^{th} position of w. This algorithm, for each word w,

Algorithm 2 input: $k \in \mathbb{N}, L \subseteq \Sigma^+$, Σ fixed, L finite, returns: the largest integer l_4 such that L satisfies $W(4, k, l_4)$

```
    Let $l_4$, if $k \geq s_L$, return.
 2: Let $M = (Q_1, \Sigma \cup \{\#\}, q_0, F_1, \delta_1) :=$ suffix_dfa($\Sigma^{-1}L$),
    Let $M^R = (Q_2, \Sigma \cup \{\#\}, p_0, F_2, \delta_2) :=$ suffix_dfa($(L\Sigma^{-1})^R$)
 4: for all $w \in L$ do
      Let $b_1, b_2$ be bit vectors of length $|w|$ all initialized to 0, let $j := 0$,
 6:   while $j \leq |w|$ do
        if $\delta(w(1) \cdots w(j)) \cap F_1 \neq \emptyset$ then
 8:       set $b_1(j) := 1$,
        end if
10:     if $\delta(w(|w|) \cdots w(|w| - j + 1) \cap F_2 \neq \emptyset$ then
          set $b_2(|w| - j + 1) := 1$,
12:     end if
        j := j+1,
14:   end while
      if there exists $j$ such that $(k \leq j) \wedge (k \leq |w| - j + 1) \wedge (b_1(j) = 1) \wedge (b_2(j+1) = 1)$
      then
16:     $l_4 := l_4 + 1$.
      end if
18: end for
```

remembers every position of w which has the prefix of that length in $<_{\text{s}}^{-1} (L)$ and it also remembers every position of w^R which has the prefix of that length in $<_{\text{s}}^{-1} (L^R)$. Then $w = uv$ for some u, v with $u \in <_{\text{s}}^{-1} (L), v^R \in <_{\text{s}}^{-1} (L^R)$ and $|u|, |v| \geq k$ if and only if statement 14 is true. Hence, upon completion, l_4 will be the largest integer such that L satisfies $W(4, k, l_4)$. Furthermore, this method also runs in $O(s_L)$ time.

Property 5 can also be verified easily. Indeed, w^R is defined if and only if

Algorithm 3 input: $k \in \mathbb{N}, L \subseteq \Sigma^+$, Σ fixed, L finite, returns: largest integer l_5 such that L satisfies $W(5, k, l_5)$

```
    Let $l_5 := 0$, if $k \geq s_L$ then return.
 2: Let $M = (Q, \Sigma \cup \{\#\}, q_0, F, \delta) :=$ suffix_dfa($L^R$),
    for all $w \in L$ do
 4:   if $\delta(q_0, w^R)$ is defined then
        let $l_5 := l_5 + 1$.
 6:   end if
    end for
```

$w^R u \in \leq_{\text{s}}^{-1} (L)$ for some u if and only if $w^R \in \leq_{\text{i}}^{-1} (L)$. Hence we can decide this property in time $O(s_L)$.

For property 6, we note that a word $w \leq_{\text{i}} v \in L^+$ if and only if $w \in R = (\leq_{\text{i}}^{-1} (L)) \cup (\leq_{\text{s}}^{-1} (L)L^* \leq_{\text{p}}^{-1} (L))$. Moreover, it is easy to construct an NFA

$M = (Q, \Sigma, q_0, F, \delta)$ accepting R in linear time, with the number of states linear in s_L. In addition, it is well-known that we can test whether a word w is in the language generated by an NFA in time $O(|Q||w|)$ (see [6]). Thus, to find the largest integer l_6 such that L satisfies $W(6, k, l)$, we construct the NFA from L and decide membership of w^R for each $w \in L$. This takes time $O(|w_1||Q| + \cdots + |w_m||Q|) = O(|Q|s_L)$. Thus, one can decide whether a finite language L satisfies $W(6, k, l)$ in time $O(s_L^2)$.

Finally, the strong properties can also be verified straightforwardly using the algorithms presented above. Indeed, they are just a special case where $l = |L|$. We summarize the preceding thusly:

Proposition 4. *Let i satisfy $1 \leq i \leq 5$ and let Σ be some fixed alphabet. Then given a finite language $L \subseteq \Sigma^+$ as input without duplicates and $k \in \mathbb{N}$, we can both find the largest integer l such that L satisfies $W(i, k, l)$ and we can decide whether L satisfies $V(i, k)$ in time $O(s_L)$. Furthermore, we can both find the largest l whereby L satisfies $W(6, k, l)$ and we can decide whether L satisfies $V(6, k)$ in time $O(s_L^2)$.*

5 Conclusions and Discussion

We have presented here a formalization of the process of gene compression that occurs in many viral genomes. We have shown dependencies and relationships between these properties and demonstrated that, in general, most of the weak versions of the properties can be decided for languages defined by nondeterministic finite automata augmented with reversal-bounded counters while the strong versions can be decided for regular languages. Most significantly, we have given algorithms which can efficiently decide these properties for real viral genomes and provide information which is immediately useful to virologists.

These algorithms give us the ability to study the relative amount of gene compression between related viruses in a quantifiable way. It may be possible to infer evolutionary relationships between viruses using this information. The fact that genes overlap one another provides a very serious constraint for viral genome evolution. It is known that viruses occasionally aquire genes horizontally (that is, a gene from an infected host becomes part of the virus's own genome). Clearly, only those genes which meet very specific constraints (*e.g.* those that are "compressible" relative to the virus's genome) will be able to be incorporated into the virus. Using the algorithms presented here and real viral genome data, we can find target genes in the host organism which, due to their structure, have the greatest probability of being incorporated into the viral genome.

Finally, the formal properties here also present a framework for automated classification of a virus given only its genome. The family of Coronaviruses, for example, has a very regular genomic structure: a single strand of +-sense RNA of length 27-30kb. The beginning of this RNA strand always encodes a viral polymerase (often as part of a polyprotein) and the remainder encodes a series of "nested" genes. Each of these nested genes is a proper suffix of the previous gene.

This structure can obviously be formally encoded using the properties given here. Similar compression regularities can be found in other viral genomes and encoded using our properties. Classification of a new virus is then simply a matter of verifying compliance to our properties and then checking to see if this matches any known structures.

By formalizing this ancient form of data compression, we have provided tools which will allow for further insight in the molecular evolution of viruses and assist in the automated classification of new viruses by reference to only their genomes.

References

1. J. Berstel. *Transductions and Context-Free Languages.* B.B. Teubner, Stuttgart, 1979.
2. A. Blumer, J. Blumer, M.T. Chen, A. Ehrenfeucht, Haussler D., and J. Seiferas. The smallest automaton recognizing the subwords of a text. *Theoretical Computer Science*, 40(1):31–55, 1985.
3. A.J. Cann. *Principles of Molecular Virology, 3e.* Academic Press, San Diego, CA, 2001.
4. S. Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages.* North-Holland Publishing Company, Amsterdam, 1975.
5. S. Ginsburg and E.H. Spanier. Bounded algol-like languages. *Transactions of the American Mathematical Society*, 113(2):333–368, 1964.
6. J. Holub and B. Melichar. Implementation of nondeterministic finite automata for approximate pattern matching. In J.-M. Champarnaud, D. Maurel, and D. Ziadi, editors, *WIA '98, Lecture Notes in Computer Science*, volume 1660, pages 92–99. Springer-Verlag, 1999.
7. O. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1):116–133, 1978.
8. D.C. Krakauer. Evolutionary principles of genome compression. *Comments on Theoretical Biology*, 7(4):215–236, 2002.
9. A. Salomaa. *Formal Languages.* Academic Press, New York, 1973.
10. E.K. Wagner and M.J. Hewlett. *Basic Virology.* Blackwell Science, Malden, MA, 1999.