Theoretical and computational properties of transpositions *

Mark Daley[1,2], Ian McQuillan[2,†], James M. McQuillan[3], Kalpana Mahalingam[4]

[1] Department of Computer Science, Department of Biology
University of Western Ontario
London, ON N6A 5B7, Canada
daley@csd.uwo.ca

[2] Department of Computer Science
University of Saskatchewan
Saskatoon, SK, S7N 5A9, Canada
mcquillan@cs.usask.ca

[3] Department of Computer Science
Western Illinois University
Macomb, IL, 61455-1390, USA
jm-mcquillan@wiu.edu

[4] Department of Computer Science
Indian Institute of Technology Madras
Chennai - 600 036, India
kmahalingham@iitm.ac.in

Abstract

Transposable genetic elements are prevalent across many living organisms from bacteria to large mammals. Given the linear primary structure of genetic material, this process is natural to study from a theoretical perspective using formal language theory. We abstract the process of genetic transposition to operations on languages and study it combinatorially and computationally. It is shown that the power of such systems is large relative to the classic Chomsky Hierarchy. However, we are still able to algorithmically determine whether or not a string is a possible product of the iterated application of the operations.

Key words: bioinformatics, transposable elements, transpositions, formal language theory, mathematical modelling

*Published in Natural Computing, 10, 795–804. https://doi.org/10.1007/s11047-010-9207-z
†Corresponding author, daley@csd.uwo.ca, ph: +519-661-3566, fax: +519-661-3515.

1 Introduction

There are many different types of changes which can occur to a genome throughout evolution. These include mutations, and small insertions and deletions. In addition, there is a large class of DNA sequences which can move from one location to another within a genome. These are collectively known as *transposable elements* or *transpositions*. They are extremely important biologically, as they are estimated to occupy between $64\% - 73\%$ of corn [1]. Moreover, the dispersed repetitive fraction of the human genome is estimated to be 46% [2].

See [3] for a good survey of transpositions. There are two main classes of transposable elements. Class I are referred to as *retroelements*. These elements are produced by using reverse transcription (which allows DNA to be produced from RNA) to make copies of themselves. The new DNA then integrates at a new location of the genome. Intuitively, this class of transpositions operates similar to a *copy-and-paste* mechanism. Class II are known as DNA transposons. They can operate with either a copy-and-paste or *cut-and-paste* mechanism. Furthermore, many types of transposable elements have a target site preference, which will affect the contexts around the DNA in which the elements will be re-inserted.

These elements are quite natural to study theoretically using formal language theory, abstracted to operations on words and languages. In this paper, we attempt to lay the foundations for the study of this operation both mathematically and algorithmically. We then analyze some basic properties of the operation.

In section 2, we give some mathematical preliminaries necessary for the rest of the paper. In section 3, we give the definitions of different types of transpositions abstracted to operations on words and languages. Sections 4 and 5 investigate the computational power of applying both classes of transpositions iteratively. Section 6 investigates the properties of each type being applied a single time. This section is useful towards the algorithmic study of transpositions, which is investigated in section 7.

We were able to demonstrate that iterated type-1 transpositions produce arbitrary Recursively Enumerable languages from finite initial and transposition languages modulo right quotient with a regular language. In addition, we were able to closely characterize the non-iterated versions of the operations. Then we found that we could algorithmically determine if a string is a possible product of either iterated type-1 or type-2 transpositions, or even both together at once, so long as we can determine membership in the initial and transposition language.

We hope that the study is useful for mathematical modelling, to improve our understanding of transpositions, and also towards bioinformatics.

2 Mathematical Preliminaries

Let $\mathbb{N}$ be the set of positive integers and let $\mathbb{N}_0$ be the set of nonnegative integers.

We refer to [4] for language theory preliminaries. Let Σ be a finite alphabet. We denote, by Σ^* and Σ^+, the sets of all words and non-empty words, respectively, over Σ and the empty word by λ. A language L is any subset of Σ^*. Let $L, R \subseteq \Sigma^*$. We denote by $R^{-1}L = \{z \in \Sigma^* \mid yz \in L \text{ for some } y \in R\}$ and $LR^{-1} = \{z \in \Sigma^* \mid zy \in L \text{ for some } y \in R\}$.

Let $x \in \Sigma^*$. We let $|x|$ denote the length of x. For each $a \in \Sigma$, we let $|x|_a$ be the number of occurrences of a in x. If $y = (x_1, \ldots, x_n)$ is some n-tuple over Σ^*, then $|y| = |x_1| + \cdots + |x_n|$.

For $i \in \mathbb{N}$, let $x(i)$ be a_i if $x = a_1 \cdots a_i \cdots a_n$, $a_j \in \Sigma, 1 \leq j \leq n$, and undefined otherwise.

For $n \in \mathbb{N}_0$, let $\Sigma^n = \{w \in \Sigma^* \mid |w| = n\}$, $\Sigma^{\geq n} = \{w \in \Sigma^* \mid |w| \geq n\}$ and $\Sigma^{\leq n} = \{w \in \Sigma^* \mid |w| \leq n\}$.

u is copied and pasted between x,y, yielding:

Figure 1: We show a diagram illustrating a transposition of type 1.

u is cut and pasted between x,y, yielding:

Figure 2: We show a diagram illustration a transposition of type 2.

Let $x, y \in \Sigma^*$. We say x is a *prefix* of y, written $x \leq_{\mathrm{p}} y$, if $y = xu$, for some $u \in \Sigma^*$. We say x is a *suffix* of y, written $x \leq_{\mathrm{s}} y$, if $y = ux$, for some $u \in \Sigma^*$. We say x is an *infix* of y, written $x \leq_{\mathrm{i}} y$, if $y = uxv$, for some $u, v \in \Sigma^*$.

For each word $x \in \Sigma^*$ with $\Sigma = \{a_1, \ldots, a_n\}$ using some fixed ordering, we associate its *Parikh vector*, $p_\Sigma(x)$ by $p_\Sigma(x) = (|x|_{a_1}, |x|_{a_2}, \ldots, |x|_{a_n})$. We extend p_Σ to languages $L \subseteq \Sigma^*$ by $p_\Sigma(L) = \bigcup_{x \in L} p_\Sigma(x)$. We omit Σ if it is understood. Given Σ and Parikh vector v, the *Parikh inverse set* of v, $p^{-1}(v)$, is $p^{-1}(v) = \{x \in \Sigma^* \mid p(x) = v\}$. If $x \in \Sigma^*$ and $L \subseteq \Sigma^*$, then let $\mathrm{perm}(x) = \{y \in \Sigma^* \mid p_\Sigma(y) = p_\Sigma(x)\}$ and $\mathrm{perm}(L) = \bigcup_{x \in L} \mathrm{perm}(x)$. Note that $p^{-1}(p(x)) = \mathrm{perm}(x)$ for all $x \in \Sigma^*$.

A set of vectors is called *semilinear* if it can be represented as a union of a finite number of sets of the form $\{v_0 + \sum_{i=1}^{m} \alpha_i v_i \mid \alpha_i \in \mathbb{N}_0 \text{ for } 1 \leq i \leq m\}$ where $v_i \in \mathbb{N}_0^n$ for $0 \leq i \leq m$, for some dimension n. A language L is called *semilinear* if the set $p_\Sigma(L)$ is a semilinear set. Two languages L, L' are called *letter-equivalent* if and only if $p_\Sigma(L) = p_\Sigma(L')$. Thus, if two languages are letter equivalent, then one is semilinear if and only if the other one is also. It is known that a language L is semilinear if and only if it is letter-equivalent to a regular language [5].

A directed graph is a tuple $G = (V, E)$ in the usual way. For $v \in V$, we let $C(v)$ be the connected component of v viewed as a subgraph of G.

3 Transpositions

Abstracting from the biology of Class I and Class II transpositions to the core transformations of cut-and-paste and copy-and-paste, we define two types of formal transposition as follows. Let Σ be a finite alphabet. A 4-tuple $t = (x, u, y, z)$ is a *transposition* if $x, y \in \Sigma^*$, $u \in \Sigma^+$, and $z \in \{1, 2\}$. The transposition t is *type-1* if $z = 1$; it is *type-2* if $z = 2$.

The *transposition-concatenation map* $h^c : \Sigma^* \times \Sigma^+ \times \Sigma^* \times \{1, 2\} \to \Sigma^+$ is defined as $h^c(x, u, y, z) = xuy$, where $t = (x, u, y, z)$ is a transposition. Also, we define h_i^c to be the i^{th} coordinate of the transposition, for $1 \leq i \leq 4$, and $h^c(T) = \{h^c(t) \mid t \in T\}$. We also define $T^{\leq n} = \{t \in T \mid |h^c(t)| \leq n\}$.

A *transposition schema* is an ordered pair $\gamma = (\Sigma, T)$ where Σ is an alphabet and T is a set of transpositions.

Given a transposition schema $\gamma = (\Sigma, T)$ and $r, s \in \Sigma^*$, we say that r *is T-translatable to* s (or s *is T-translatable from* r), denoted $r \to_T s$ if at least one of the following two conditions is true:

1. there exists $(x, u, y, 1) \in T$ such that $r = \alpha u \beta = x'xyy'$ and $s = x'xuyy'$, for some $\alpha, \beta, x', y' \in \Sigma^*$,
2. there exists $(x, u, y, 2) \in T$ such that $r = \alpha u \beta$, $\alpha\beta = x'xyy'$, $s = x'xuyy'$, for some $\alpha, \beta, x', y' \in \Sigma^*$.

We abbreviate $r \to_T s$ with $r \to s$ when T is understood. We say that $r \to_T^+ s$ if $r = r_1 \to_T \cdots \to_T r_k = s$, for some $r_1, \ldots, r_k, k \geq 2$. We say that $r \to_T^* s$ if $r \to_T^+ s$ or $r = s$.

Intuitively, $r \to_T s$ implies that applying some transposition in T to r can produce s. For a type-1 transposition $(x, u, y, 1)$ to be applied, u must appear in r, and then is pasted between sites x and y. This is pictured in Figure 1. This closely reflects the copy-and-paste functionality of class I transposable elements. Notice here that it is possible for u to overlap with xy. For example, if $r = \alpha u_1 a_1 a_2 a_3 a_4 a_5 a_6 u_2 \beta$, with $u = u_1 a_1 a_2 a_3 a_4 a_5 a_6 u_2, x = a_1 a_2 a_3, y = a_4 a_5 a_6, a_i \in \Sigma$ and $(x, u, y, 1) \in T$, then $r \to_T s = \alpha u_1 a_1 a_2 a_3 u_1 a_1 a_2 a_3 a_4 a_5 a_6 u_2 a_4 a_5 a_6 u_2 \beta$. And indeed, biologically it is possible for transpositions to insert into other transpositions throughout evolution [6]. With type-2, u gets cut out of r before it is pasted between x and y. This is pictured in Figure 2

Let $\gamma = (\Sigma, T)$ be a transposition schema and let $L \subseteq \Sigma^*$. We define the *non-iterated transposition of* L to be $\gamma(L) = \{s \mid r \to s,\ r \in L\}$. Furthermore, we define the following inductively,

$$\begin{aligned}
\gamma^0(L) &= L, \\
\gamma^{i+1}(L) &= \gamma(\gamma^i(L)), \text{ for } i \geq 0, \\
\gamma^*(L) &= \bigcup_{i \geq 0} \gamma^i(L), \\
\gamma^+(L) &= \bigcup_{i > 0} \gamma^i(L), \\
\gamma^{\leq i}(L) &= \bigcup_{0 \leq j \leq i} \gamma^j(L),
\end{aligned}$$

where $\gamma^*(L)$ is referred to as the *iterated transposition of* L.

We say a transposition schema is a *type-1* (respectively *type-2*) transposition schema if all transpositions are of type-1 (respectively type-2). In this case, we identify the transpositions as ordered triples and omit the final coordinate. We say a set of transpositions has *finite contexts* if $\{u_1 u_3 \mid (u_1, u_2, u_3) \in T\}$ is finite. We say the set has *finite appearances* if $\{u_2 \mid (u_1, u_2, u_3) \in T\}$ is finite.

In order to discuss T as belonging to a language family, we will often write T as being a subset of $\#\Sigma^*\$\Sigma^*\$\Sigma^*\#$ by identifying $t = (x, u, v)$ in T as $t = \#x\$u\$y\#$ where \$ and # are any symbols not in Σ.

Note that $\gamma^*(L) = \gamma^+(L) \cup \{L\}$ in the above definition. Also, note that T is finite if and only if T has both finite contexts and finite appearances. Intuitively, $\gamma^*(L)$ will consist of all strings which can be produced after applying any arbitrary number of transpositions to any string in L. Note that "nested transpositions", or transpositions being inserted into another transposition have been found to occur [7].

4 Iterated type-1 transpositions

The contextual "copy-and-paste" nature of schemata restricted to type-1 transpositions suggests a natural relationship with the pure insertion grammars. Pure grammars have no nonterminal symbols and thus all words derivable from a finite set of axioms using a finite set of rules belong to the language generated by the pure grammar.

Proposition 4.1 *Let $L \subseteq \Sigma^*$ be a language generated by a pure insertion grammar $G = (\Sigma, A, P)$. There exist finite languages $L', T \subseteq \Sigma^*$ and a type-1 transposition schema $\gamma = (\Sigma \cup \{\$\}, T)$ such that $\gamma^*(L')(\$\Sigma^*)^{-1} = L$ where $\$ \notin \Sigma$.*

Proof. We recall the definition of a pure insertion grammar [8], as a triple $G = (\Sigma, A, P)$ where Σ is a finite alphabet, $A \subseteq \Sigma^*$ is a finite set of axioms and $P \subseteq \Sigma^* \times \Sigma^* \times \Sigma^*$ is a finite set of insertion rules. The derivation relation ($\Rightarrow$) for a pure insertion grammar is defined such that for $u, v \in \Sigma^*$, $u \Rightarrow v$ iff $u = u'xyu''$, $v = u'xwyu''$, for some $(x, w, y) \in P$, $u, u'' \in \Sigma^*$. The language generated by such a grammar is defined in the usual way.

Given a pure insertion grammar G, we denote by w_P a word consisting of the concatenation of the second component of all rules in P with an arbitrary, but fixed, ordering on P. Formally, $w_P = \prod_{p \in P} \pi_2(p)$, where π is the projection function.

We now construct a finite language $L' \subseteq \Sigma \cup \{\$\}$ and a type-1 transposition schema $\gamma = (\Sigma \cup \{\$\}, T)$ as follows: for all $w \in A$, we include $w\$w_P \in L'$, as A is finite, so must be L'; for all rules $(x, u, y) \in P$ we include $(x, u, y) \in T$ which, again, must be finite.

It is true that $L(G) \subseteq \gamma^*(L')(\$w_p)^{-1}$, as each derivation applied in G can be simulated by the corresponding transposition in γ as the second coordinate of the transposition will appear in w_p. Thus, $L(G) \subseteq \gamma^*(L')(\$\Sigma^*)^{-1}$. It is clear that $\gamma^*(L')(\$\Sigma^*)^{-1} \subseteq L(G)$, as any transposition applied acts on either the part of the word before \$, or after \$. If it gets applied before the \$, then the segment before \$ is in $L(G)$ (and indeed the same is true if it gets applied after the \$). ■

Notice that the \$ symbol is needed if one wants to perform a right quotient with Σ^* as not every pure insertion grammar generates a language closed under taking suffixes. However, if we weaken the statement by taking right quotient with regular languages instead of Σ^*, then the \$ symbol is no longer necessary in the statement. The following corollaries are now immediate from [8], [9].

Corollary 4.1 *For every recursively enumerable language L there exist homomorphisms h and g, a finite language L' and a finite type-1 transposition schema γ such that $g(h^{-1}(\gamma^*(L')(\$\Sigma^*)^{-1})) = L$. There exist also a regular language R, a finite language L'' and a finite type-1 transposition schema γ_1 such that $\gamma_1^*(L'')(R)^{-1} = L$.*

Corollary 4.2 *There exist non-semilinear languages which can be generated by a finite type-1 transposition schema acting on a finite language.*

5 Iterated type-2 transpositions

In this section, we will explore basic mathematical properties and computational capacity of iterated type-2 transpositions.

The following are immediate from the definitions of type-2 transposition systems:

Lemma 5.1 *Let $\gamma = (\Sigma, T)$ be a type-2 transposition schema with $L \subseteq \Sigma^*$. Then the following are true:*

1. $p_\Sigma(L) = p_\Sigma(\gamma^*(L))$,

2. $\gamma^*(L) \subseteq \mathrm{perm}(L)$,

3. $\gamma^*(L)$ *is semilinear if and only if* L *is semilinear,*

4. $w \in \gamma^*(L)$ *if and only if* $w \in \bar{\gamma}^*(L')$ *where* $\bar{\gamma} = (\Sigma, T^{\leq |w|})$ *and* $L' = \mathrm{perm}(w) \cap L$.

5. *If* L *is finite, then* $\gamma^*(L) = \bar{\gamma}^*(L)$, *where* $\bar{\gamma} = (\Sigma, T^{\leq n}), n = \max\{|v| \mid v \in L\}$

Proof. The first part is immediate as $r \to s$ implies $p_\Sigma(r) = p_\Sigma(s)$, and the second and third parts follow from the first. We will prove the fourth statement as follows:

"$\Rightarrow$" Assume $w \in \gamma^*(L)$. It follows from part (2) that it is enough to use L'. There exist $v_0, \ldots, v_n \in \gamma^*(L)$ and $t_1, \ldots, t_n \in T$ such that $v_i \to_{\{t_{i+1}\}} v_{i+1}$ for every $0 \leq i < n$ where $v_0 \in L$. In particular, in order for each t_i to be used, by the definition of a transposition system, $|t_i| \leq |w|$.

"$\Leftarrow$" immediate.

The fifth statement follows from the fourth. ■

Thus, in terms of generative power, if L is finite, we can assume T is also, by ignoring rules of T which are too long to be used.

Lemma 5.2 *Let* $\Gamma = (\Sigma, T)$ *be a type-2 transposition schema with* $L \subseteq \Sigma^*$ *and* $\{(\lambda, a, \lambda) \mid a \in \Sigma\} \subseteq T$. *Then* $\gamma^*(L) = \mathrm{perm}(L)$.

Proof. "$\subseteq$" This follows from Lemma 5.1 (2).

"$\supseteq$" This follows as, for any $w \in L$, we can move every letter $a \in \Sigma$ to any position of w. Thus, $\mathrm{perm}(w) \subseteq \gamma^*(L)$, and hence $\mathrm{perm}(L) \subseteq \gamma^*(L)$. ■

The following is now immediate, since we know that the families of regular and context-free languages are not closed under permutation.

Corollary 5.1 *The families of regular and context-free languages are not closed under type-2 transpositions with finite languages. Indeed they are not closed under transposition languages of size* $|\Sigma|$.

Here, as is customary in formal language theory, a family of languages $\mathcal{L}_1$ is closed under transpositions from $\mathcal{L}_2$ if the non-iterated transposition of a language in $\mathcal{L}_1$ with a transposition set in $\mathcal{L}_2$ always yields a (usually different) language in $\mathcal{L}_1$.

We will revisit both iterated type-1 and -2 schemas in section 7.

6 Non-iterated transpositions

In this section, we will explore the power of applying the transformations a single time. These results will become important for the next section which studies the operations algorithmically.

First we will see that under common formal language theoretic operations, closure under iterated transpositions implies closure under non-iterated transpositions.

Proposition 6.1 *Let* $z \in \{1, 2\}$. *Let* $\mathcal{L}_1$ *be a language family closed under* λ*-free homomorphism, inverse homomorphism and intersection with regular languages, and let* $\mathcal{L}_2$ *be closed under inverse homomorphism and intersection with regular languages. If* $\mathcal{L}_1$ *is closed under iterated type-z transpositions from* $\mathcal{L}_2$, *then* $\mathcal{L}_1$ *is closed under non-iterated type-z transpositions from* $\mathcal{L}_2$.

Proof. First we will consider type-1. Let $L \in \mathcal{L}_1$, let $\gamma = (\Sigma, T)$ be a transposition schema, $T \in \mathcal{L}_2$, $T \subseteq \#\Sigma^*\$\Sigma^+\$\Sigma^*\#$, and let h be a homomorphism from $(\Sigma \cup \{\alpha\})^*$ to Σ^*, where α is a new symbol, defined by $h(a) = a$, for each $a \in \Sigma$ and $h(\alpha) = \lambda$. Let $R = (\alpha\Sigma)^*\alpha$, a regular language, and let $L' = h^{-1}(L) \cap R$. Thus, each word of L has α inserted between every two letters. Let $\bar{T} = h^{-1}(T) \cap \#(\alpha\Sigma)^*\alpha\$(\alpha\Sigma)^+\$(\Sigma\alpha)^*\#$, and let $\bar{\gamma} = (\Sigma \cup \{\alpha\}, \bar{T})$. Then, it is evident that $\bar{\gamma}^*(L') \cap ((\alpha\Sigma)^*\alpha\alpha(\Sigma\alpha)^* \cup (\alpha\Sigma)^*\alpha\alpha(\Sigma\alpha)^*\Sigma\Sigma\alpha(\Sigma\alpha)^*) = \bar{\gamma}(L')$ because any word in $\bar{\gamma}^2(L') - \bar{\gamma}(L')$ would produce either three consecutive α's, or two sections of two α's (words in $(\alpha\Sigma)^*\alpha\alpha(\Sigma\alpha)^*$ are produced via transpositions with an empty third component and words in $(\alpha\Sigma)^*\alpha\alpha(\Sigma\alpha)^*\Sigma\Sigma\alpha(\Sigma\alpha)^*$ are produced via transpositions with a non-empty third component). Furthermore, $h(\bar{\gamma}(L')) = \gamma(L)$, and every language family closed under λ-free homomorphism, inverse homomorphism and intersection with regular languages is also closed under linear-erasing homomorphisms. Thus type-1 follows.

The same proof works identically with type-2. ■

Although the conditions of the proposition are abstract, most nondeterministic machines defining families of languages, including each family from the Chomsky Hierarchy are closed under these operations.

Proposition 6.2 *Let $z \in \{1, 2\}$. The family of counter languages (and the context-free languages) are not closed under non-iterated type-z regular transpositions with contexts of size* 0. *Furthermore, the same is true with iterated transpositions.*

Proof. We will start with type-1 transpositions. Assume otherwise. Let $L = \{a^nqa^nq \mid n \geq 0\}$, and let $T = \{(\lambda, qa^nq, \lambda) \mid n \geq 0\}$, and let $\gamma = (\{a, q\}, T)$ be a transposition system. Then consider $\gamma(L) \cap a^*qa^*qqa^*q = \gamma^*(L) \cap a^*qa^*qqa^*q = \{a^nqa^nqqa^nq \mid n \geq 0\}$, which isn't a counter language, a contradiction.

Next, we consider type-2 transpositions. Assume otherwise. Let $L = \{a^nqpa^npa^mqa^m \mid n, m \geq 0\}$. Let $T = (\lambda, pa^np, \lambda) \mid n \geq 0\}$, and let $\gamma = (\{p, q, a\}, T)$ be a transposition system. Then, $\gamma(L) \cap a^*qa^*qpa^*pa^* = \gamma^*(L) \cap a^*qa^*qpa^*pa^* = \{a^nqa^mqpa^npa^m \mid n, m \geq 0\}$, which isn't a counter language, contradiction. ■

We see however, that for the non-iterated version, it is the unbounded appearances which make the difference. Similarly to Proposition 6.1, the conditions are satisfied by all standard families of languages defined by nondeterministic machines.

Proposition 6.3 *Let $\mathcal{L}$ be a language family closed under inverse homomorphism, λ-free homomorphism and intersection with regular languages. Then the following are true:*

1. *$\mathcal{L}$ is closed under regular type-1 transpositions with finite appearances,*
2. *If $\mathcal{L}$ is closed under arbitrary homomorphism, then $\mathcal{L}$ is closed under regular transpositions*[1] *with finite appearances.*

Proof. We will start by proving the second statement, restricted to type-2 transpositions, as both statements are easy variants of this. Let $\gamma = (\Sigma, T)$ be a transposition system where T is a regular language (which for simplicity's sake, we will denote as being a subset of $\#\Sigma^*\$\Sigma^*\$\Sigma^*\#$), with finite appearances. Let $M = (Q, \Sigma, q_0, F, \delta)$ be a deterministic finite automaton accepting T. Let $L \in \mathcal{L}, L \subseteq \Sigma^*$. Let $X = \{h_2^c(t) \mid t \in T\}$, which is finite, and for each $x \in X$, let $Q(x) =$

[1] This can include both type-1 and type-2 transpositions.

$\{(q_1, q_2, q_3, q_4) \mid \#u_1\$x\$u_3\# \in T, \delta(q_0, \#) = q_1, \delta(q_1, u_1) = q_2, \delta(q_2, \$x\$) = q_3, \delta(q_3, u_3) = q_4\}$. Each of these sets is finite.

Let $\bar{\Sigma} = \{\bar{a} \mid a \in \Sigma\}$, and let h be a homomorphism from $(\Sigma \cup \bar{\Sigma})^*$ to Σ^* defined by $h(a) = h(\bar{a}) = a$, for each $a \in \Sigma$. Let $\bar{X}$ be the barred version of X and let $L' = h^{-1}(L) \cap \Sigma^* \bar{X} \Sigma^* \in \mathcal{L}$.

Next, we construct a nondeterministic gsm K which operates on L' as follows: first, on input $u_1u_2u_3$, with $u_1u_3 \in \Sigma^*$, and $u_2 \in \bar{\Sigma}^*$, K nondeterministically guesses $x \in X$, and $(q_1, q_2, q_3, q_4) \in Q(x)$. This is done by adding a λ transition from the initial state q_0 to any state $q_{x,z}$, where $x \in X, z \in Q(x)$. Then, in parallel it does the following

1. verifies $h(u_2) = x$ and erases u_2,
2. while reading $u_1u_3 = a_1 \cdots a_m$ (ie. ignoring barred letters), guesses positions $i, j, k, 1 \leq i \leq j < k$ and verifies that $\delta(q_1, a_i \cdots a_j) = q_2$, then outputs x, then verifies $\delta(q_3, a_{j+1} \cdots a_k) = q_4$.

We will describe how to implement condition 2, and it is clear that we can easily add in 1. We use states $(q^1_{x,z}, q), q^2_{x,z}, (q^3_{x,z}, q), q_f$ for all $q \in Q$ (Q is the state set for the finite automaton). There are transitions from $q_{x,z}$ on each letter a while outputting a, then there is a transition from $q_{x,z}$ to $(q^1_{x,z}, q_1)$ applied nondeterministically on λ input, then it transitions from $(q^1_{x,z}, p)$ to $(q^1_{x,z}, q)$ on a letter a, while outputting a, if $q \in \delta(p, a)$. Then, from $(q^1_{x,z}, q_2)$, it can transition nondeterministically on λ input to $q^2_{x,z}$ while outputing x. Then, it switches to state $(q^3_{x,z}, q_3)$ and (similar to state $(q^1_{x,z}, q)$) simulates M with the second coordinate until hitting $(q^3_{x,z}, q_4)$, at which point it switches to q_f and outputs the input.

Then $K(L') = \gamma(L)$ and as every language family satisfying the assumptions is closed under nondeterministic gsms, it follows that $K(L') \in \mathcal{L}$.

We shall next deal with type-1 transpositions. If in the proof above, we modify K so that it does not erase u_2, but rather outputs $h(u_2) = x$, then K is λ-free, and $K(L') = \gamma(L)$.

Lastly, if T consists of both type-1 and type-2 transpositions, then the nondeterministic gsm can guess at the beginning of its computation. ■

7 Membership of transpositions

We now discuss the algorithmic problem of deciding whether a given string is a possible product of iterated transpositions. We will start with type-1 only.

Proposition 7.1 *Let $\gamma = (\Sigma, T)$ be a type-1 transposition schema with $L \subseteq \Sigma^*$ and $w \in \Sigma^*$. Then $w \in \gamma^*(L)$ if and only if $w \in \bar{\gamma}^{\leq |w|}(L')$ where $\bar{\gamma} = (\Sigma, T^{\leq |w|})$ and $L' = L \cap \Sigma^{\leq |w|}$.*

Proof. "⇒" If $w \in \gamma^*(L)$, then $w \in \gamma^m(v)$ for some $m \geq 0$ minimal and $v \in L$. Thus, $|w| \geq |v| + m$, which makes $m \leq |w|$ and $|v| \leq |w|$.

"⇐" immediate. ■

We see that there is an algorithm as long as we can determine if a given string is in L and T.

Proposition 7.2 *Let $\gamma = (\Sigma, T)$ be a type-1 transposition schema with $L \subseteq \Sigma^*$ where we can decide membership in L and T and let $w \in \Sigma^*$. Then we can decide whether $w \in \gamma^+(L)$.*

Proof. By Proposition 7.1, it suffices to decide if $w \in \bar{\gamma}^{\leq |w|}(L')$ where $\bar{\gamma} = (\Sigma, T^{\leq |w|})$ and $L' = L \cap \Sigma^{\leq |w|}$. As L and T have a decidable membership problem, we can effectively construct both L' and $T^{\leq |w|}$. As L' and $T^{\leq |w|}$ are both finite, it follows that we can effectively construct the finite language $\bar{\gamma}^{\leq |w|}(L')$ and then we can test if w is in this language. ■

Next, we will study the same question for type-2 transpostions.

Let $\gamma = (\Sigma, T)$ be a type-2 transposition schema and let $L \subseteq \Sigma^*$. We wish to find an algorithm that, given $w \in \Sigma^*$, will decide whether or not $w \in \gamma^+(L)$. We solve the problem first if L and T are finite.

Proposition 7.3 *Let $\gamma = (\Sigma, T)$ be a type-2 transposition schema and $L \subseteq \Sigma^*$ where L and T are finite and effectively given, and let $w \in \Sigma^*$. Then there is an algorithm which can determine if $w \in \gamma^*(L)$.*

Proof. It is immediate from the definition that $\gamma^{i+1}(L) = \gamma^i(L)$ implies $\gamma^*(L) = \gamma^i(L)$. Since γ only permutes, $\gamma^i(L)$ is finite for all i and there exists i such that $\gamma^i(L) = \gamma^*(L)$ is finite and thus we can decide whether $w \in \gamma^*(L)$. ■

More generally, we can decide membership in $\gamma^+(L)$ as long as we can within L and T.

Proposition 7.4 *Let $\gamma = (\Sigma, T)$ be a type-2 transposition schema, let $L \subseteq \Sigma^*$ where we can decide membership in L and T and let $w \in \Sigma^*$. Then it is decidable whether $w \in \gamma^*(L)$.*

Proof. Indeed, as we can decide membership in L and T, it is possible to construct $L \cap \text{perm}(w)$ and $T^{\leq |w|}$ by testing whether v is in L, for each $v \in \text{perm}(w)$ and by testing whether x is in T, for each $|x| \leq |w|$. The proposition then follows from Proposition 7.3. ■

Lastly, we will use these results to decide membership in $\gamma^+(L)$, even if γ contains both type-1 and -2 transpositions.

Proposition 7.5 *Let $\gamma = (\Sigma, T)$ be a transposition schema (potentially containing both type-1 and -2 rules), let $L \subseteq \Sigma^*$ where we can decide membership in L and T and let $w \in \Sigma^*$. Then it is decidable whether $w \in \gamma^*(L)$.*

Proof. Let T_1 (respectively T_2) be the subset of T with type-1 (respectively type-2) rules. Let $\gamma_1 = (\Sigma, T_1)$ and $\gamma_2 = (\Sigma, T_2)$.

Let $L_1 = L \cap \Sigma$ and for all $n \geq 1$, let

$$L_{n+1} = (\gamma_2^*(\gamma_1(L_n) \cup (L \cap \Sigma^{n+1})) \cap \Sigma^{\leq n+1}) \cup L_n.$$

We will show by induction that for all $n \geq 1$, $L_n = \gamma^*(L) \cap \Sigma^{\leq n}$.

It is clear that $L_1 = \gamma^*(L) \cap \Sigma^{\leq 1}$. Let $k \geq 1$ and assume $L_k = \gamma^*(L) \cap \Sigma^{\leq k}$. It is immediate that $L_{k+1} \subseteq \gamma^*(L) \cap \Sigma^{\leq k+1}$. Let $w \in \gamma^*(L) \cap \Sigma^{k+1}$. Thus, either $v \to v' \to^* w$ where $v \in L_k = \gamma^*(L) \cap \Sigma^{\leq k}$ and $v' \notin L_k$ (as L_k only consists of words of length less than or equal to k), or $v \to^* w$ where $v \in L \cap \Sigma^{k+1}$ (depending upon whether w is obtained with at least one type-1 transposition or only type-2). Assume the first case. Then $|v| < |v'| = |w|$, and thus $|v'|$ must be obtained via one application of a rule from γ_1 followed by zero or more from γ_2. Thus, $w \in L_{k+1}$. Assume the second case. Then w must be obtained from v via zero or more applications of γ_2 from $L \cap \Sigma^{k+1}$ and thus $w \in L_{k+1}$. Hence, by induction, $L_n = \gamma^*(L) \cap \Sigma^{\leq n}$, for all $n \geq 1$.

Finally, to test whether $w \in \gamma^*(L)$, it suffices to check whether $\gamma^*(L) \cap \Sigma^{\leq |w|}$. To start, we can construct the finite languages $L' = L \cap \Sigma^{\leq |w|}$ and $T^{\leq |w|}$ by deciding membership in L and T. Then, we can iteratively construct $L_1, \ldots, L_{|w|}$ as follows: at iteration $i+1$, we add in all words in L of length $i+1$, all words of length $i+1$ obtained via one application of a type-1 rule from L_i, and then all words obtained via iterated type-2 transpositions via these new words which we can determine by Proposition 7.4. Thus, we can decide if $w \in L_{|w|} = \gamma^*(L) \cap \Sigma^{\leq |w|}$ and if $w \in \gamma^*(L)$.

■

8 Conclusions

We have abstracted the process of both classes of transposable elements to operations on strings and languages. We investigated some basic mathematical properties and the computational power of transpositions. In particular, it was shown that we can generate arbitrary recursively enumerable languages from finite initial and finite transposition languages, modulo right quotient by a regular set. Moreover, non-semilinear languages can be generated similarly. For type-2 transpositions, we can generate only semilinear languages, but the operation is strictly more powerful than permutation. Algorithmically, it was demonstrated that we can decide membership after application of both iterated type-1 and iterated type-2 transpositions, so long as we can decide membership in both the initial and transposition languages. Then, we were able to use these results to show decidability even when both type-1 and -2 transpositions were present simultaneously.

Future work will consider the time complexity necessary to determine this and other decision questions. Furthermore, we would like to study these complexity questions under certain realistic assumptions, in an attempt for the algorithms to be useful from the perspective of bioinformatics and the analysis of data.

Acknowledgements

This research was supported by grants from the Natural Sciences and Engineering Research Council of Canada, institutional grants of the University of Saskatchewan and the University of Western Ontario and the SHARCNET Research Chairs Program.

References

[1] Meyers, B., Tingey, S., Morgante, M.: Abundance, distribution, and transcriptional activity of repetitive elements in the maize genome. Genome Res. **11** (2001) 1660–1676

[2] Consortium, I.H.G.S.: Initial sequencing and analysis of the human genome. Nature **409** (2001) 860–921

[3] Kidwell, M.G.: Transposable elements. In Gregory, T.R., ed.: The Evolution of The Genome. Elsevier Academic Press (2005)

[4] Salomaa, A.: Formal Languages. Academic Press, New York (1973)

[5] Harrison, M.A.: Introduction to Formal Language Theory. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1978)

[6] Giordano, J., Ge, Y., Gelfand, Y., Abrusán, G., Benson, G., Warburton, P.E.: Evolutionary history of mammalian transposons determined by genome-wide defragmentation. PLoS Computational Biology **3**(7) (2007) 1321–1334

[7] Quesneville, H., Bergman, C.M., Andrieu, O., Autard, D., Nouaud, D., Ashburner, M., Anxolabehere, D.: Combined evidence annotation of transposable elements in genome sequences. PLoS Computational Biology **1**(2) (2005) 166–175

[8] Kari, L., Sosík, P.: On the weight of universal insertion grammars. Theoretical Computer Science (396) (2008) 264–270

[9] Martin-Vide, C., Păun, G., Salomaa, A.: A characterization of recursively enumerable languages by means of insertion grammars. Theoretical Computer Science **205** (1998) 195–205