Transducer Descriptions of DNA Code Properties and Undecidability of Antimorphic Problems

Lila Kari[1], Stavros Konstantinidis[2], and Steffen Kopecki[1,2]

[1] The University of Western Ontario, London, Ontario, Canada
lila@csd.uwo.ca, steffen@csd.uwo.ca
[2] Saint Mary's University, Halifax, Nova Scotia, Canada
s.konstantinidis@smu.ca

Abstract. This work concerns formal descriptions of *DNA* code properties and related (un)decidability questions. This line of research allows us to give a property as input to an algorithm, in addition to any regular language, which can then answer questions about the language and the property. Here we define DNA code properties via transducers and show that this method is strictly more expressive than that of regular trajectories, without sacrificing the efficiency of deciding the satisfaction question. We also show that the maximality question can be undecidable. Our undecidability results hold not only for the fixed DNA involution but also for any fixed antimorphic permutation. Moreover, we also show the undecidability of the antimorphic version of the Post Corresponding Problem, for any fixed antimorphic permutation.

Keywords: codes, DNA properties, trajectories, transducers, undecidability

1 Introduction

The study of *formal* methods for describing independent language properties (widely known as code properties) provides tools that allow one to give a property as input to an algorithm and answer questions about this property. Examples of such properties include *classic* ones [4, 17, 27, 28] like prefix codes, bifix codes, and various error-detecting languages, as well as DNA code properties [2, 10, 11, 13–15, 18, 20–22, 25] like θ-nonoverlapping and θ-compliant languages. A formal description method should be expressive enough to allow one to describe many desirable properties. Examples of formal methods for describing *classic* code properties are the implicational conditions method of [16], the trajectories method of [5], and the transducer methods of [8]. The latter two have been implemented to some extent in the Python package FAdo [9]. A formal method for describing DNA code properties is the method of trajectory DNA code properties [6, 22].

Typical questions about properties are the following:

Satisfaction problem: given the description of a property and the description of a regular language, decide whether the language satisfies the property.
Maximality problem: given the description of a property and the description of a regular language that satisfies the property, decide whether the language is maximal with respect to the given property.
Construction problem: given the description of a property and a positive integer n, find a language of n words (if possible) satisfying the given property.

In the above problems regular languages are described via (non-deterministic) finite automata (NFA). Depending on the context, properties are described via trajectory regular expressions or transducer expressions. The satisfaction problem is the most basic one and can be answered usually in efficient polynomial time. The maximality problem as stated above can be decidable, in which case it is normally PSPACE-hard. For existing transducer and trajectory properties, both problems can be answered using the online (formal) language server LaSer [24], which relies on FAdo. For the construction problem a simple statistical algorithm is included in FAdo, but we think that this problem is far from being well-understood.

The *general objective* of this research is to develop methods for formally describing DNA code properties that would allow one to express various combinations of such properties and be able to get answers to questions about these properties in an actual implementation. The contributions of this work are as follows:

1. The definition of a new simple formal method for describing many DNA code properties, called θ-*transducer properties*, some of which *cannot* be described by the existing transducer and trajectory methods for classic code properties; see Sect. 3.
2. The demonstration that the new method of transducer DNA code properties is properly more expressive than the method of trajectories; see Sect. 4.
3. The demonstration that the maximality problem can be decidable for some transducer DNA code properties but undecidable for some others; see Sect. 5.
4. The demonstration that some classic undecidable problems (like PCP) remain undecidable when rephrased in terms of *any fixed* (anti-)morphic permutation θ of the alphabet, with the case $\theta = \mathrm{id}$ corresponding to these classic problems, where id is the *(morphic) identity*; see Sect. 6.

Even though, our main motivation is the description of DNA-related properties, we follow the more general approach which considers properties described by transducers involving a fixed (anti-)morphic permutation θ; again, the classical transducer properties are obtained by letting $\theta = \mathrm{id}$.

This is a condensed conference version which does not contain full proofs of our results. The full version of this paper, containing all proofs, can be accessed on arXiv [19]. It also contains additional examples of DNA properties which can be described by θ-transducer properties: these properties naturally extend the hierarchy of DNA properties that is used in [13, 18, 20].

2 Basic Notions and Background Information

In this section we lay down our notation for formal languages, (anti-)morphic permutations, transducers, and language properties. We assume the reader to be familiar with the fundamental concepts of language theory; see e. g., [12,26]. Then, in Sect. 2.2 we recall the method of transducers for describing classic code properties, and in Sect. 2.3 we recall the method of trajectories for describing DNA-related properties.

2.1 Formal Languages and (Anti-)morphic Permutations

For an alphabet A and a language L over A we have the notation: $A^+ = A^* \setminus \{\varepsilon\}$, where ε is the empty word; and $L^c = A^* \setminus L$. For an integer $k \geq 2$ we define the *generic alphabet* $A_k = \{0, 1, \ldots, k-1\}$ *of size* k. Throughout this paper we only consider alphabets with at least two letters because our investigations would become trivial over unary alphabets.

Let $w \in A^*$ be a word. Unless confusion arises, by w we also denote the singleton language $\{w\}$, e. g., $L \cup w$ means $L \cup \{w\}$. If $w = xyz$ for some $x, y, z \in A^*$, then x, y, and z are called *prefix*, *infix* (or *factor*), and *suffix* of w, respectively. For a language $L \subseteq A^*$, the set $\mathrm{Pref}(L) = \{x \in A^* \mid \exists y \in A^* \colon xy \in L\}$ denotes the language containing all prefixes of words in L. If $w = a_1 a_2 \cdots a_n$ for letters $a_1, a_2, \ldots, a_n \in A$, then $|w| = n$ is the *length of* w; for $b \in A$, $|w|_b = |\{i \mid a_i = b, 1 \leq i \leq n\}|$ is the tally of b occurring in w; the i-th letter of w is $w_{[i]} = a_i$ for $1 \leq i \leq n$; the infix of w from the i-th letter to the j-th letter is $w_{[i;j]} = a_i a_{i+1} \cdots a_j$ for $1 \leq i \leq j \leq n$; and the *reverse* of w is $w^R = a_n a_{n-1} \cdots a_1$.

Consider a generic alphabet A_k with $k \geq 2$. The *identity* function on A_k is denoted by id_k; when the alphabet is clear from the context, the index k is omitted. For a *permutation* (or bijection) $\theta \colon A_k \to A_k$, and for $i \in \mathbb{Z}$, the permutation θ^i is the *i-fold composition* of θ; i. e., $\theta^0 = \mathrm{id}_k$, $\theta^i = \theta \circ \theta^{i-1}$, and $\theta^{-i} = (\theta^i)^{-1} = (\theta^{-1})^i$ for $i > 0$. An *involution* θ is a permutation such that $\theta = \theta^{-1}$.

A permutation θ over A_k can naturally be extended to operate on words in A_k^* as (a) *morphic permutation* $\theta(uv) = \theta(u)\theta(v)$, or (b) *antimorphic permutation* $\theta(uv) = \theta(v)\theta(u)$, for $u, v \in A_k^*$. As before, the inverse θ^{-1} of the (anti-)morphic permutation θ over A_k^* is the (anti-)morphic extension of the permutation θ^{-1} over A_k^*. The identity id_k always denotes the morphic extension of id_k while the antimorphic extension of id_k, called the *mirror image* or reverse, is usually denoted by the exponent ^{R}.

Example 1. The *DNA involution*, denoted as δ, is an antimorphic involution on $\Delta = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ such that $\delta(\mathtt{A}) = \mathtt{T}$ and $\delta(\mathtt{C}) = \mathtt{G}$, which implies $\delta(\mathtt{T}) = \mathtt{A}$ and $\delta(\mathtt{G}) = \mathtt{C}$.

2.2 Describing Classic Code Properties by Transducers

A *(language) property* $\mathcal{P}$ is any set of languages. A language L *satisfies* $\mathcal{P}$, or *has* $\mathcal{P}$, if $L \in \mathcal{P}$. Here by a property $\mathcal{P}$ we mean an *(n-)independence* in the sense of [17]: there exists $n \in \mathbb{N} \cup \{\aleph_0\}$ such that a language L satisfies $\mathcal{P}$ if and only if all nonempty subsets $L' \subseteq L$ of cardinality less than n satisfy $\mathcal{P}$. A language L satisfying $\mathcal{P}$ is *maximal* (with respect to $\mathcal{P}$) if for every word $w \in L^c$ we have $L \cup w$ does not satisfy $\mathcal{P}$—note that, for any independence $\mathcal{P}$, every language in $\mathcal{P}$ is a subset of a maximal language in $\mathcal{P}$ [17]. As we shall see further below the focus of this work is on 3-independence properties that can also be viewed as independent with respect to a binary relation in the sense of [28].

A *transducer* $\mathbf{t}$ is a non-deterministic finite state automaton with output; see e. g., [3,30]. Here we only consider transducers whose input and output alphabets are equal: a transducer is a quintuple $\mathbf{t} = (Q, A, E, I, F)$, where A is the input and output alphabet, Q is a finite set of states, E is a set of directed edges between states from Q which are labeled by word pairs $(u, v) \in A^* \times A^*$, I is a set of initial states, and F a set of final states. If $\mathbf{t}$ realizes (x, y) then we write $y \in \mathbf{t}(x)$. We say that the set $\mathbf{t}(x)$ contains all possible outputs of $\mathbf{t}$ on input x. The transducer $\mathbf{t}^{-1}$ is the inverse of $\mathbf{t}$; that is, $x \in \mathbf{t}^{-1}(y)$ if and only if $y \in \mathbf{t}(x)$ for all words x, y. Let θ be an (anti-)morphic permutation and $\mathbf{t}$ be a transducer which are both defined over the same alphabet A. The transducer $\mathbf{t}$ is called *θ-input-preserving* if for all $w \in A^+$ we have $\theta(w) \in \mathbf{t}(w)$; $\mathbf{t}$ is called *θ-input-altering* if for all $w \in A^+$ we have $\theta(w) \notin \mathbf{t}(w)$. We use the simpler terms *input-preserving* and *input-altering* $\mathbf{t}$, respectively, when $\theta = \mathrm{id}$. Note that $\theta(w) \in \mathbf{t}(w)$ is equivalent to $w \in \theta^{-1}(\mathbf{t}(w))$ as well as $\mathbf{t}^{-1}(\theta(w)) \ni w$.

Definition 1 ([8]). *An input-altering transducer* $\mathbf{t}$ describes the property *that consists of all languages L such that*

$$\mathbf{t}(L) \cap L = \emptyset. \tag{1}$$

An input-preserving transducer $\mathbf{t}$ describes the property *that consists of all languages L such that*

$$w \notin \mathbf{t}(L \setminus w), \quad \textit{for all } w \in L. \tag{2}$$

A property is called an input-altering (resp. input-preserving) transducer property, if it is described by an input-altering (resp. input-preserving) transducer.

Note that every input-altering transducer property is also an input-preserving transducer property. Input-altering transducers can be used to describe properties like prefix codes, bifix codes, and hypercodes. Input-preserving transducers are intended for error-detecting properties, where in fact the transducer plays the role of the communication channel.

Many input-altering transducer properties can be described in a simpler manner by *trajectory regular expressions* [5,8], that is, regular expressions over {0, 1}. For example, the expression 0^*1^* describes prefix codes and the expression $1^*0^*1^*$ describes infix codes. On the other hand, there are natural transducer properties that cannot be described by trajectory expressions [8].

2.3 Describing DNA-related Properties by Trajectories

In [2, 10, 11, 13–15, 18, 20–22, 25] the authors consider numerous properties of languages inspired by reliability issues in DNA computing. We state three of these properties below. Let θ be an antimorphic permutation over A_k^*. Recall that, in the DNA setting, $\theta = \delta$ is an involution, and therefore, we have $\theta^2 = \mathrm{id}$.

(A) A language L is *θ-nonoverlapping* if $L \cap \theta(L) = \emptyset$.

(B) L is *θ-compliant* if $\forall w \in \theta(L), x, y \in A_k^* \colon xwy \in L \implies xy = \varepsilon$.

(C) L is *strictly θ-compliant* if it is θ-nonoverlapping and θ-compliant.

Many of the existing DNA-related properties can be modelled using the concept of a bond-free property, first defined in [22] and later rephrased in [6] in terms of trajectories. We follow the fomulation in [6]. Let $\bar{e}_1$ and $\bar{e}_2$ be two regular trajectory expressions. First, we define the following language operators.

$$\Phi_{\bar{e}_1,\bar{e}_2}(L) = (((L \rightsquigarrow_{\bar{e}_1} A^+) \cap A^+) \sqcup\!\sqcup_{\bar{e}_2} A^*) \cup (((L \rightsquigarrow_{\bar{e}_1} A^*) \cap A^+) \sqcup\!\sqcup_{\bar{e}_2} A^+), \quad (3)$$

$$\Phi^{\mathbf{s}}_{\bar{e}_1,\bar{e}_2}(L) = ((L \rightsquigarrow_{\bar{e}_1} A^*) \cap A^+) \sqcup\!\sqcup_{\bar{e}_2} A^*. \quad (4)$$

The language operations $\sqcup\!\sqcup_{\bar{a}}$ and $\rightsquigarrow_{\bar{a}}$ are *shuffle* (or scattered insertion) and *scattered deletion*, respectively, over the set of trajectories $\bar{a}$; see [6,23] for details.

Definition 2. *([6]) Let θ be an involution (or more generally a permutation) and $\bar{e}_1, \bar{e}_2$ be two regular trajectory expressions. The* bond-free property described by $(\bar{e}_1, \bar{e}_2)$ *is*

$$\mathcal{B}_\theta(\bar{e}_1, \bar{e}_2) = \{L \subseteq A^* \mid \theta(L) \cap \Phi_{\bar{e}_1,\bar{e}_2}(L) = \emptyset\}. \quad (5)$$

The strictly bond-free property described by $(\bar{e}_1, \bar{e}_2)$ *is*

$$\mathcal{B}^{\mathbf{s}}_\theta(\bar{e}_1, \bar{e}_2) = \{L \subseteq A^* \mid \theta(L) \cap \Phi^{\mathbf{s}}_{\bar{e}_1,\bar{e}_2}(L) = \emptyset\}. \quad (6)$$

A regular θ-trajectory property *is a bond-free property described by $(\bar{e}_1, \bar{e}_2)$, or a strictly bond-free property described by $(\bar{e}_1, \bar{e}_2)$, for some pair $(\bar{e}_1, \bar{e}_2)$.*

3 New Transducer-based DNA-related Properties

A question that arises from the discussion in sections 2.2 and 2.3 is whether existing transducer-based properties include DNA-related properties. It turns out that this is not the case; see Proposition 1. In this section, we define new transducer-based properties that are appropriate for DNA-related applications, we demonstrate Proposition 1, and discuss how existing DNA-related properties can be described with transducers.

Definition 3. *A transducer $\mathbf{t}$ and an (anti-)morphic permutation θ, defined over the same alphabet, describe 3-independent properties in two ways:*
1.) strict θ-transducer property ($\mathcal{S}$-property)*: L satisfies the property $\mathcal{S}_{\theta,\mathbf{t}}$ if*

$$\theta(L) \cap \mathbf{t}(L) = \emptyset \quad (7)$$

2.) weak θ-transducer property ($\mathcal{W}$-property)*: L satisfies the property $\mathcal{W}_{\theta,\mathbf{t}}$ if*

$$\forall w \in L\colon \theta(w) \notin \mathbf{t}(L \setminus w) \tag{8}$$

Any of the properties $\mathcal{S}_{\theta,\mathbf{t}}$ or $\mathcal{W}_{\theta,\mathbf{t}}$ is called a θ-transducer property*.*

The difference between $\mathcal{S}$-properties and $\mathcal{W}$-properties is that $\mathcal{S}_{\theta,\mathbf{t}}$ includes no language containing a word w such that $\theta(w) \in \mathbf{t}(w)$, while this case is allowed for some $L \in \mathcal{W}_{\theta,\mathbf{t}}$. For fixed $\mathbf{t}$, θ, and L, Condition (7) implies that for all $w \in L$ we have $\theta(w) \cap \mathbf{t}(L \setminus w) = \emptyset$ which is equivalent to Condition (8). In other words, if L satisfies $\mathcal{S}_{\theta,\mathbf{t}}$, then L satisfies $\mathcal{W}_{\theta,\mathbf{t}}$ as well. If $\theta = \mathrm{id}$ and $\mathbf{t}$ is input-altering, or input-preserving, then the above defined properties specialize to the existing ones stated in Definition 1.

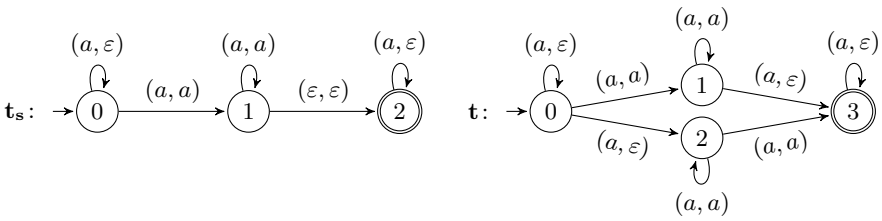

Fig. 1. Together with θ, the left transducer describes the strictly θ-compliant property and the right one describes the θ-compliant property. See Example 2 for explanations.

Example 2. In Fig. 1, an arrow with label (a, a) represents a set of edges with labels (a, a) for all $a \in A$; and similarly for an arrow with label (a, ε). For any word xwy, the left transducer $\mathbf{t_s}$ can delete x, then keep w (which has to be non-empty), and then delete y. Thus, $\mathbf{t_s}(L) \cap \theta(L) = \emptyset$ if and only if L is strictly θ-compliant. Now let xwy with $xy \neq \varepsilon$ and $w \neq \varepsilon$. If y is nonempty, the right transducer $\mathbf{t}$ can delete x, then keep w, and then delete y using the upper path (containing state 1); and if x is nonempty, $\mathbf{t}$ can delete x, then keep w, and then delete y using the lower path (containing state 2). Thus, $\mathbf{t}(L) \cap \theta(L) = \emptyset$ if and only if L is θ-compliant. Using FAdo [9] format the left transducer can be specified by the following string, assuming alphabet {`a`, `b`}

```
@Transducer 2 * 0\n0 a @epsilon 0\n0 b @epsilon 0\n0 a a 1\n
0 b b 1\n1 a a 1\n1 b b 1\n1 @epsilon @epsilon 2\n2 a @epsilon 2\n
2 b @epsilon 2\n
```

The next result demonstrates that existing transducer properties are not suitable for describing even simple DNA-related properties.

Proposition 1. *The δ-nonoverlapping property is not describable by any input-preserving transducer.*

4 Expressiveness of Transducer-based Properties

In this section we examine the descriptive power of the newly defined transducer DNA-related properties, that is, the θ-transducer properties. In Theorem 1 we show that these properties properly include the regular θ-trajectory properties. On the other hand, in Proposition 2 we show that there is an independent DNA-related property that is not a θ-transducer property.

Proposition 2. *The θ-free property (defined below) [13] is not a θ-transducer property.*

(D) A language $L \subseteq A^*$ is θ-free *if and only if $L^2 \cap A^+\theta(L)A^+ = \emptyset$.*

The following DNA language property is considered in Theorem 1

$$\mathcal{H} = \{L \subseteq \Delta^* \mid H(u, \delta(v)) \geq 2, \text{ for all } u, v \in L\},$$

where $H(\cdot, \cdot)$ is the Hamming distance function with the assumption that its value is ∞ when applied on different length words. Note that $\mathcal{H}$ is described by δ and the transducer shown in Fig. 2.

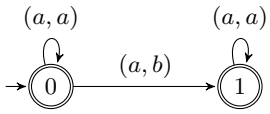

Fig. 2. The transducer $\mathbf{t}$ describing, together with δ, the $\mathcal{S}$-property $\mathcal{H}$: the displayed transducer $\mathbf{t}$ realizes (u, v) if and only if $H(u, v) < 2$; therefore, $\delta(L) \cap \mathbf{t}(L) = \emptyset$ if and only if $H(u, \delta(v)) \geq 2$ for all $u, v \in L$.

Example 3. The DNA language $L_1 = \{\texttt{AGG}, \texttt{CCA}\}$ does not satisfy $\mathcal{H}$ because $H(\texttt{CCA}, \delta(\texttt{AGG})) = 1$. The DNA language $L_2 = \{\texttt{AAA}, \texttt{CCT}\}$ satisfies $\mathcal{H}$ because $\delta(\texttt{AAA}) = \texttt{TTT}$ and all words $u \in L_2$ contain at most one $\texttt{T}$.

Theorem 1.

1. *Let θ be an antimorphic involution. Every regular θ-trajectory property is a θ-transducer property (in particular an $\mathcal{S}$-property).*
2. *Property $\mathcal{H}$ is a δ-transducer property, but not a (regular) δ-trajectory one.*

5 The Satisfaction and Maximality Problems

For $\theta = \mathrm{id}$ and for input-altering and -preserving transducers the satisfaction and maximality problems are decidable [8]. In particular, for a regular language L given via an automaton $\mathbf{a}$, Condition (1) can be decided in time $\mathcal{O}(|\mathbf{t}||\mathbf{a}|^2)$, where the function $|\cdot|$ returns the size of the machine in question (= number of states

plus number of edges plus the length of all labels on the edges). Condition (2) can be decided in time $\mathcal{O}(|\mathbf{t}||\mathbf{a}|^2)$, as noted in Remark 1 below. The maximality problem is decidable, but PSPACE-hard, for both input-altering and -preserving transducer properties.

Remark 1. Let $\mathbf{s} = \mathbf{t} \downarrow \mathbf{a} \uparrow \mathbf{a}$ be the transducer obtained by two product constructions: first on the input of $\mathbf{t}$ with $\mathbf{a}$; then, on the output of the resulting transducer with $\mathbf{a}$. In [8] the authors suggest to decide whether or not L satisfies the input-preserving transducer property $\mathcal{W}_{\mathrm{id},\mathbf{t}}$ by testing if the transducer $\mathbf{s}$ is functional. However, deciding $L \in \mathcal{W}_{\mathrm{id},\mathbf{t}}$ can be done by the cheaper test of whether or not $\mathbf{s}$ implements a (partial) identity function. Using the identity test from [1], we obtain that Condition (2) can be decided in time $\mathcal{O}(|\mathbf{t}||\mathbf{a}|^2)$ when the alphabet is considered constant. Also note that the identity test does not require that $\mathbf{t}$ is input-preserving if $\theta = \mathrm{id}$. When θ is antimorphic, however, the identity test does not work anymore and we have to resort to the more expensive functionality test for θ-input-preserving transducers.

In this work we are interested in the case when $\theta \neq \mathrm{id}$ is antimorphic; furthermore, the θ-input-altering or -preserving restrictions on the transducer are not necessarily present in the definition of $\mathcal{W}$-properties or $\mathcal{S}$-properties. Table 1 summarizes under which conditions the satisfaction and maximality problems are decidable for regular languages.

Problem	Property $\mathcal{S}_{\theta,\mathbf{t}}$		Property $\mathcal{W}_{\theta,\mathbf{t}}$	
	no restriction	$\mathbf{t}$ is θ-i.-altering	no restriction	$\mathbf{t}$ is θ-i.-preserving
Satisfaction	decidable in $\mathcal{O}(\|\mathbf{t}\|\|\mathbf{a}\|^2)$ as in [8]		decidable Theorem 2	decidable in $\mathcal{O}(\|\mathbf{t}\|^2\|\mathbf{a}\|^4)$ as in [8]
Maximality	undecidable Corollary 2	decidable, PSPACE-hard Theorem 3, Corollary 1		

Table 1. (Un-)decidability of the satisfaction and the maximality problems for a fixed antimorphic permutation θ, a given transducer $\mathbf{t}$, and a regular language L given via an automaton $\mathbf{a}$.

Remark 2. We note that deciding the satisfaction question for any θ-trajectory property involves testing the emptiness conditions in (5) or (6), which requires time $\mathcal{O}(|\mathbf{a}|^2|\mathbf{a}_1||\mathbf{a}_2|)$, where $\mathbf{a}_1, \mathbf{a}_2$ are automata corresponding to $\bar{e}_1, \bar{e}_2$. Such a property can be expressed as θ-transducer $\mathcal{S}$-property (recall Theorem 1) using a transducer of size $\mathcal{O}(|\mathbf{a}_1||\mathbf{a}_2|)$ and, therefore, the satisfaction question can still be solved within the same asymptotic time complexity.

5.1 The Satisfaction Problem for non-restricted $\mathcal{W}$-properties

We establish the decidability of non-restricted transducer $\mathcal{W}$-properties for regular languages. We do not concern the complexity of this algorithm; optimizing

the algorithm and analyzing its complexity is part of future research. Let $\mathbf{t}$ be a transducer, θ be an antimorphic permutation, and L be a regular language over the alphabet A. Let $\mathbf{a}_L$ and $\mathbf{a}_{\theta(L)}$ be the NFAs accepting the languages L and $\theta(L)$, respectively. Let $\mathbf{s} = \mathbf{t} \downarrow \mathbf{a}_L \uparrow \mathbf{a}_{\theta(L)}$ be the product transducer such that $y \in \mathbf{s}(x)$ if and only if $y \in \mathbf{t}(x)$, $x \in L$, and $y \in \theta(L)$.

Let $T_{\mathbf{s}} = \left\{(x_1, x_2, x_3) \in (A^*)^3 \;\middle|\; |x_1x_2x_3| \leq |\mathbf{s}|\right\}$ be a set of word triples. Note that the length restrictions for the words ensures that $T_{\mathbf{s}}$ is a finite set. For each triple $t = (x_1, x_2, x_3) \in T_{\mathbf{s}}$ we define a relation

$$R_t = \left\{(x_1(x_2)^k x_3, \theta(x_1(x_2)^k x_3)) \;\middle|\; k \in \mathbb{N}\right\} \subseteq A^* \times A^*.$$

Lemma 1. *The regular language L satisfies $\mathcal{W}_{\theta,\mathbf{t}}$ if and only if the relation realized by $\mathbf{s}$ is included in $\bigcup_{t \in T_{\mathbf{s}}} R_t$.*

The inclusion in Lemma 1 is decidable performing the following two tests: 1.) verify that $\mathbf{s} \subseteq \bigcup_{(x_1,x_2,x_3) \in T_s} (x_1 x_2^* x_3) \times \theta(x_1 x_2^* x_3)$; and 2.) verify that $|x| = |y|$ for all pairs (x, y) that label an accepting path in $\mathbf{s}$. Note that the inclusion test can be performed because the right-hand-side relation is recognizable [3]. The second test follows the same ideas as the algorithm outlined in [1] which decides whether or not a transducer implements a partial identity function.

Theorem 2. *Let L be a regular language given as automaton, $\mathbf{t}$ be a given transducer, and θ be a given antimorphic involution (all defined over A). It is decidable whether L satisfies $\mathcal{W}_{\theta,\mathbf{t}}$ or not.*

5.2 The Maximality Problem

Here we show how to decide maximality of a regular language L with respect to a θ-transducer property; see Theorem 3. This result only holds when we consider $\mathcal{W}$-properties or when we consider $\mathcal{S}$-properties for θ-input-altering transducers. As in the case of existing transducer properties, it turns out that the maximality problem is PSPACE-hard; see Corollary 1. When we consider general $\mathcal{S}$-properties, the maximality problem becomes undecidable; see Corollary 2.

Theorem 3. *For an antimorphic permutation θ, a transducer $\mathbf{t}$, and a regular language L, all defined over A_k^*, such that either $L \in \mathcal{W}_{\theta,\mathbf{t}}$, or $L \in \mathcal{S}_{\theta,\mathbf{t}}$ and $\mathbf{t}$ is θ-input altering, we have that L is maximal with property $\mathcal{W}_{\theta,\mathbf{t}}$ (resp., $\mathcal{S}_{\theta,\mathbf{t}}$) if and only if*

$$L \cup \theta^{-1}(\mathbf{t}(L)) \cup \mathbf{t}^{-1}(\theta(L)) = A_k^*. \tag{9}$$

We note that it is PSPACE-hard to decide whether or not Equation (9) holds when L is given as NFA because it is PSPACE-hard to decide universality of a regular language given as NFA ($L \subseteq A_k^*$ is universal if $L = A_k^*$) [29].

Corollary 1. *For an antimorphic permutation θ, a transducer $\mathbf{t}$, and a regular language L given as NFA, all defined over A_k^*, such that either $L \in \mathcal{W}_{\theta,\mathbf{t}}$, or $L \in \mathcal{S}_{\theta,\mathbf{t}}$ and $\mathbf{t}$ is θ-input altering, we have that it is* PSPACE*-hard to decide whether or not L is maximal with property $\mathcal{W}_{\theta,\mathbf{t}}$ (resp., $\mathcal{S}_{\theta,\mathbf{t}}$).*

In the rest of this section we show that it is undecidable whether or not a transducer is θ-input-preserving. This question relates directly to the maximality problem of the empty language $\emptyset$ with respect to the property $\mathcal{S}_{\theta,\mathbf{t}}$, as stated in Corollary 2. The following Theorem can be proven using a reduction from the famous, undecidable Post correspondence problem (PCP) to the problem of deciding whether a given transducer is θ-input-preserving or not.

Theorem 4. *For every fixed antimorphic permutation θ over A_k^* with $k \geq 2$ it is undecidable whether or not a given transducer is θ-input-preserving.*

This leads to the undecidability of the maximality problem of a regular language L with respect to a θ-transducer-property $\mathcal{S}_{\theta,\mathbf{t}}$.

Corollary 2. *For every fixed antimorphic permutation θ over A_k^* with $k \geq 2$, it is undecidable whether or not the empty language $\emptyset$ is maximal with respect to the property $\mathcal{S}_{\theta,\mathbf{t}}$, for a given transducer $\mathbf{t}$.*

Note that a singleton language $\{w\}$ satisfies $\mathcal{S}_{\theta,\mathbf{t}}$ if and only if $\theta(w) \notin \mathbf{t}(w)$. Thus, the corollary follows because $\emptyset$ is maximal with property $\mathcal{S}_{\theta,\mathbf{t}}$ if and only if $\mathbf{t}$ is θ-input-preserving.

6 Undecidability of the θ-PCP and the θ-input-altering Transducer Problem

Analogous to the undecidable PCP we introduce the θ version of the PCP and prove that it is undecidable as well; see Theorem 5. Further, we utilize the θ version of the PCP in order to show that it is undecidable whether or not a transducer is θ-input-altering; see Corollary 3.

Definition 4. *For a fixed antimorphic permutation θ over A_k^*, we introduce the* θ-Post correspondence problem *(θ-PCP): given words $\alpha_0, \alpha_1, \ldots, \alpha_{\ell-1} \in A_k^+$ and $\beta_0, \beta_1, \ldots, \beta_{\ell-1} \in A_k^+$, decide whether or not there exists a non-empty sequence of integers $i_1, \ldots, i_n \in A_\ell = \{0, 1, \ldots, \ell-1\}$ such that*

$$\alpha_{i_1}\alpha_{i_2}\cdots\alpha_{i_n} = \theta(\beta_{i_1}\beta_{i_2}\cdots\beta_{i_n}).$$

Theorem 5. *For every fixed antimorphic permutation θ over A_k^* with $k \geq 2$ the θ-PCP is undecidable.*

We can utilize the θ-PCP in order to prove that it is undecidable whether or not a transducer is θ-input-altering, even for one-state transducers.

Corollary 3. *For every fixed antimorphic permutation θ over A_k^* with $k \geq 2$ it is undecidable whether or not a given (one-state) transducer is θ-input-altering.*

Corollary 3 follows because the θ-PCP instance $\alpha_0, \ldots, \alpha_{\ell-1}, \beta_0, \ldots, \beta_{\ell-1}$ has a solution if and only if the the one-state transducer $\mathbf{t}$ with edges $q \xrightarrow{(\alpha_i, \theta^2(\beta_i))} q$ for $i = 0, \ldots, \ell-1$ is not θ-input-altering.

7 Conclusions

We have defined a transducer-based method for describing DNA code properties which is strictly more expressive than the trajectory method. In doing so, the satisfaction question remains efficiently decidable. The maximality question for some types of properties is decidable, but it is undecidable for others. While some versions of the maximality question for trajectory properties are decidable, the case of any given pair of regular trajectories and any given regular language is not addressed in [6], so we consider this to be an interesting problem to solve.

The maximality questions are phrased in terms of any fixed antimorphic permutation. This direction of generalizing decision questions is also applied to the classic Post Correspondence Problem, where we demonstrate that it remains undecidable. A consequence of this is that the question of whether a given transducer is θ-input-altering is also undecidable. It is interesting to note that if, instead of fixing θ, we fix the transducer $\mathbf{t}$ to be the identity, or the transducer defining the $\mathcal{S}$-property $\mathcal{H}$ (see Fig. 2 in Sect. 4), then the question of whether or not $\theta(L) \cap \mathbf{t}(L) = \emptyset$ is decidable (given any regular language L and antimorphic permutation θ).

The topic of studying description methods for code properties requires further attention. One important aim is the actual implementation of the algorithms, as it is already done for several classic code properties [9, 24]. An immediate plan is to incorporate in those implementations what we know about DNA code properties. Another aim is to increase the expressive power of our description methods. The formal method of [16] is quite expressive, using a certain type of first order formulae to describe properties. It could perhaps be further worked out in a way that some of these formulae can be mapped to transducers. We also note that if the defining method is too expressive then even the satisfaction problem could become undecidable; see for example the method of multiple sets of trajectories in [7].

References

1. Allauzen, C., Mohri, M.: Efficient algorithms for testing the twins property. Journal of Automata, Languages and Combinatorics 8(2), 117–144 (2003)
2. Baum, E.: DNA sequences useful for computation. In: 2nd DIMACS Workshop on DNA-based computers, pp. 122–127. Princeton University (1996)
3. Berstel, J.: Transductions and Context-Free Languages. B.G. Teubner, Stuttgart (1979)
4. Berstel, J., Perrin, D., Reutenauer, C.: Codes and Automata. Cambridge University Press (2009)
5. Domaratzki, M.: Trajectory-based codes. Acta Informatica 40, 491–527 (2004)
6. Domaratzki, M.: Bond-free DNA language classes. Natural Computing 6, 371–402 (2007)
7. Domaratzki, M., Salomaa, K.: Codes defined by multiple sets of trajectories. Theoretical Computer Science 366, 182–193 (2006)
8. Dudzinski, K., Konstantinidis, S.: Formal descriptions of code properties: decidability, complexity, implementation. IJFCS 23:1, 67–85 (2012)

9. FAdo: Tools for formal languages manipulation, URL address: http://fado.dcc.fc.up.pt/ Accessed in February, 2015
10. Fan, C.M., Wang, J.T., Huang, C.C.: Some properties of involution binary relations. Acta Informatica DOI 10.1007/s00236-014-0208-8 (2014)
11. Genova, D., Mahalingam, K.: Generating DNA code words using forbidding and enforcing systems. In: Theory and Practice of Natural Computing, pp. 376–393. LNCS 7505, Springer-Verlag (2012)
12. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (1979)
13. Hussini, S., Kari, L., Konstantinidis, S.: Coding properties of DNA languages. Theoretical Computer Science 290, 1557–1579 (2003)
14. Jonoska, N., Kari, L., Mahalingam, K.: Involution solid and join codes. Fundamenta Informaticae 86, 127–142 (2008)
15. Jonoska, N., Mahalingam, K., Chen, J.: Involution codes: with application to DNA coded languages. Natural Computing 4, 141–162 (2005)
16. Jürgensen, H.: Syntactic monoids of codes. Acta Cybernetica 14, 117–133 (1999)
17. Jürgensen, H., Konstantinidis, S.: Codes. In: Rozenberg and Salomaa [26], pp. 511–607
18. Kari, L., Kitto, R., Thierrin, G.: Codes, involutions, and DNA encodings. In: Formal and Natural Computing, pp. 376–393. Springer (2002)
19. Kari, L., Konstantinidis, S., Kopecki, S.: Transducer descriptions of DNA code properties and undecidability of antimorphic problems. arXiv preprint arXiv:1503.00035 (2015)
20. Kari, L., Konstantinidis, S., Losseva, E., Wozniak, G.: Sticky-free and overhang-free DNA languages. Acta Informatica 40, 119–157 (2003)
21. Kari, L., Konstantinidis, S., Sosík, P.: Bond-free languages: formalizations, maximality and construction methods. IJFCS 16, 1039–1070 (2005)
22. Kari, L., Konstantinidis, S., Sosík, P.: On properties of bond-free DNA languages. Theoretical Computer Science 334, 131–159 (2005)
23. Kari, L., Sosík, P.: Aspects of shuffle and deletion on trajectories. Theoretical Computer Science 332, 47–61 (2005)
24. LaSer: Independent LAnguage SERver, URL address: http://laser.cs.smu.ca/independence/ Accessed in February, 2015
25. Mauri, G., Ferretti, C.: Word design for molecular computing: a survey. In: DNA 9, pp. 37–47. LNCS 7505, Springer-Verlag (2004)
26. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, Vol. I. Springer-Verlag, Berlin (1997)
27. Shyr, H.: Free Monoids and Languages. Hon Min Book Company, Taichung, 2nd edn. (1991)
28. Shyr, H., Thierrin, G.: Codes and binary relations. In: Malliavin, M.P. (ed.) Séminaire d'Algèbre Paul Dubreil, Paris 1975–1976 (29ème Année). Lecture Notes in Mathematics, vol. 586, pp. 180–188 (1977)
29. Stockmeyer, L., Meyer, A.: Word problems requiring exponential time (preliminary report). In: Proceedings of the 5th annual ACM symposium on Theory of computing. pp. 1–9. ACM (1973)
30. Yu, S.: Regular languages. In: Rozenberg and Salomaa [26], pp. 41–110