Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Transducer descriptions of DNA code properties and undecidability of antimorphic problems [☆]

Lila Kari [a,b], Stavros Konstantinidis [c,*], Steffen Kopecki [b,c]

[a] School of Computer Science, University of Waterloo, Waterloo, ON, N2L 3G1, Canada
[b] Dept. Computer Science, The University of Western Ontario, London, ON, N6A 5B7, Canada
[c] Dept. Math. & Computing Sci., Saint Mary's University, 923 Robbie St, Halifax, NS, B3H 3C3, Canada

ARTICLE INFO

Article history:
Received 30 November 2015
Available online xxxx

Keywords:
Codes
DNA properties
Trajectories
Transducers
Undecidability

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 Correspondence Problem, for any fixed antimorphic permutation.

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 [2–5] like prefix codes, bifix codes, and various error-detecting languages, as well as DNA code properties [6–17] 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 [18], the trajectories method of [19], and the transducer methods of [20]. The latter two have been implemented to some extent in the Python package FAdo [21]. A formal method for describing DNA code properties is the method of trajectory DNA code properties [12,22].

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.

[☆] This paper is an expanded version of [1]. Research supported by: NSERC Discovery Grants R2824A01 (L. Kari) and 220259 (S. Konstantinidis), and University of Western Ontario Internal Grant.

[*] Corresponding author.
E-mail addresses: lila@csd.uwo.ca (L. Kari), s.konstantinidis@smu.ca (S. Konstantinidis), steffen@csd.uwo.ca (S. Kopecki).

http://dx.doi.org/10.1016/j.ic.2017.09.004
0890-5401/© 2017 Elsevier Inc. All rights reserved.

Please cite this article in press as: L. Kari et al., Transducer descriptions of DNA code properties and undecidability of antimorphic problems, Inf. Comput. (2017), http://dx.doi.org/10.1016/j.ic.2017.09.004

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 (nondeterministic) 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 [23], which relies on FAdo. LaSer allows users to enter the desired property and language, and returns either the answer in real time (online mode), or it returns a Python program that computes the desired answer if executed at the user's site (program generation mode). For the construction problem a simple statistical algorithm is included in FAdo, but we think that this problem is far from being well-understood.

When it comes to DNA code properties in the context of formal languages, there have been a few algorithms and implementations concerning specific such properties—for example [11,17,24,25]. These provide valuable insights and contribute to the maturity of the research on DNA code properties. Most of the implementations concern the construction problem for sets of words of fixed length. In [17], the authors consider the efficient implementation of two DNA code properties as well as the UD code (Unique Decipherability code) property. The topic of DNA code properties is active and relevant, as there are laboratory experiments involving computations on DNA molecules [26].

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. While the satisfaction and construction questions are important from both the theoretical and practical viewpoints, the maximality question is at least of theoretical interest and a classic problem in the theory of codes. 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. These methods are closed under intersection of code properties. This means that if two properties can be described within the method then also the combined property can be described within the method. This outcome is important as in practice it is desirable that languages satisfy more than one property.
2. The demonstration that the new method of transducer DNA code properties is properly more expressive than the method of trajectories; see Sect. 4. Also the demonstration that the satisfaction problem is decidable for all θ-transducer properties (Sect. 5) in such a way that when these properties are trajectory DNA code properties the efficiency of the satisfaction algorithm is asymptotically the same (Remark 22).
3. The demonstration that the maximality problem can be decidable for some of these properties but undecidable for some others; see Sect. 6.
4. The demonstration that some classic undecidable problems (like *PCP*, the Post Correspondence Problem) 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. 7.
5. The presentation of a natural hierarchy of DNA properties which are all θ-transducer properties; see Sect. 8. This hierarchy generalizes the concept of bond-free properties in [7–9].

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 $\theta = \mathrm{id}$. As it turns out, in the case when θ is morphic all questions that we consider in this paper can be answered analogously to the solutions for the classical case of $\theta = \mathrm{id}$. Therefore, we focus on the transducer properties involving antimorphic permutations in this paper.

2. Basic notions and background information

We assume the reader to be familiar with the fundamental concepts of language theory; see e.g., [27,28]. In Sect. 2.1–2.3, we lay down our notation for formal languages, (anti-)morphic permutations and involutions, automata and transducers, and trajectories and related word operations. In Sect. 2.4 we recall the method of transducers for describing classic code properties, and in Sect. 2.5 we recall the method of trajectories for describing DNA code properties. In Sect. 2.6, we describe what technical tools are used to establish the main results of the paper.

For a set S, we write $|S|$ to denote its cardinality. The symbols $\mathbb{N}$ and $\mathbb{N}_+$ denote, respectively, the set of nonnegative integers and the set of positive integers.

2.1. Formal languages and (anti-)morphic permutations

An *alphabet* A is a finite set of *letters*; A^* is the set of all words or strings over A; ε denotes the *empty word*; and $A^+ = A^* \setminus \{\varepsilon\}$. A *language* L over A is a subset $L \subseteq A^*$; the *complement* L^c of L is the language $A^* \setminus L$. For an integer $m \in \mathbb{N}$ we let $A^{\le m}$ denote the set of words whose length is at most m; i.e., $A^{\le m} = \bigcup_{i \le m} A^i$. The *DNA alphabet* is $\Delta = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$.