UNIVERSITY OF BUCHAREST
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Algorithmical Aspects of Some Bio Inspired Operations

PhD THESIS Overview

Supervisor:
Prof.univ.dr. Victor MITRANA

PhD Candidate:
Adrian Marius Dumitran

Bucharest
2015

Chapter 1

Overview

1.1 Introduction

In most of the thesis we discuss about three operations inspired by biological phenomena, namely, the prefix-suffix duplication, the bounded prefix-suffix duplication and the prefix-suffix-square completion operations. Duplication is one of the most frequent and less understood mutations among the genome rearrangements [24]. Roughly speaking, duplication is the process in which a stretch of DNA is duplicated yielding two or more adjacent copies. This process may happen at any position in the chromosome, including its beginning and its end. Actually, the distribution of these tandem repeats varies widely along the chromosomes and some authors consider that approximately 5% of the genome rearrangements are different types of duplications [26].

It is considered that a so-called phylogenetic analysis, might be useful in the investigation of the evolution of species, by the studying duplications along the genome the most likely duplication history could be determined [27]. The detection of these tandem repeats and algorithms for tandem repeats reconstructing history have received a great deal of attention in bioinformatics [2, 1, 25]. However, a special type of duplications, known as telomeres, appear only at the ends of chromosomes. Generally, telomeres consist of tandem repeats of a small number of nucleotides, specified by the action of telomerase. They are considered to be protective DNA-protein complexes found at the end of eukaryotic chromosomes which stabilize the linear chromosomal DNA molecule [5, 24]. The length of telomeric DNA is important for the chromosome stability: the loss of telomeric repeat sequences may result in chromosome fusion and lead to chromosome instability [23]. In [26] one states that it is a further challenge the sequencing of the 20% of the genome that is formed by repetitive heterochromatin which is implicated in the process of chromosome replication and maintenance.

The interpretation of duplications as a formal operation on words has in-

spired several works in the areas of Combinatorics on Words and Formal Languages, starting with [3, 14] and continued in [8, 28, 21]and the references therein. It is worth mentioning that a PhD thesis has been devoted to this topic [20]. These works have been the starting point for this thesis.

In [15] duplications that appear at both ends of the words namely prefix-suffix duplications were first considered. The basic motivation of introducing these operations was to mathematically model a special type of duplications within DNA sequences, that appear only at the ends of these sequences, also known as telomeres. Another motivation would be to model the process of generating long terminal repeats (LTRs): identical sequences of DNA that repeat hundreds or thousands of times at either end of some specific DNA sequence. Such sequences are used, for instance, by viruses to insert their genetic material into the host genomes.

In [15] the class of languages that can be defined by the iterative application of the prefix-suffix duplication to a word are investigated, and the class is compared to other well studied classes of languages. It is shown that the languages of this class have a rather complicated structure even if the initial word is rather simple: they are already non-context-free as soon as the initial word contains at least two different letters. Algorithms for the membership and distance computation problems are also given. In this context we considered a weaker variant of the prefix-suffix duplications, called the bounded prefix-suffix duplication. The new model allowed us to solve some problems that remained unsolved in [15] and also the new model seems closer to the biochemical reality that inspired the definition of this operation. It seems more practical and closer to the biological reality to consider that the factor added by the prefix-suffix duplication cannot be arbitrarily long.

In the same context of prefix-suffix duplication languages we introduced the prefix-suffix square completion operation [10]. The initial motivation of studying the prefix-suffix duplication were some biological processes that essentially create repetitions at the ends of the genetic sequences ; however, the formal operations defined in [15] assumed that such repetitions are created by replicating their root. In [10] we assumed a different point of view: we considered the possibility of creating squares (the simplest type of repetition) at one of the ends of the word by completing a prefix or suffix of the considered sequence to a square.

Keeping to the above mentioned operations we found it interesting to see how our operations can be used to generate infinite words [10]. We first show that the infinite Fibonacci word, the Period-doubling word and the infinite word Thue-Morse can be generated by suffix square completion. This exhibits a property that seems interesting to us: every (infinite) word generated by suffix completion contains squares, but there are (infinite) words generated by

this operation (which basically creates squares) that avoid any repetition of (rational) exponent higher than 2. In comparison, we show that the Thue-Morse infinite word cannot be generated by prefix-suffix duplication. However, we show that one can generate an infinite cube-free word by suffix-duplication. This is a weaker version of the result obtained for square completion: every (infinite) word generated by suffix duplication contains squares, but there are (infinite) words generated by this operation that avoid any repetition of integer exponent higher than 2.

We then proceed to investigate the problem of efficiently detecting the existence of repetitive structures occurring at both ends of some sequence. For instance, words that do not end or start with repetitions may model DNA sequences that went through some degenerative process that destroyed the terminal repeats, affecting their stability or functionalities. Moving away from the biological motivation, words that do not start nor end with repetitions seem to be interesting from a combinatorial point of view, as well. Indeed, repetitions-free words (i.e., words that do not contain consecutive occurrences of the same factors) are central in combinatorics on words, stringology, and their applications (see, e.g., [22, 16]); words that do not have repetitive prefixes or suffixes model a weaker, but strongly related, notion. In [11] and in this thesis we show how to efficiently compute the number of prefix-suffix-square free factors in a word, how to count them but also how to compute the longest one.

Keeping to our biological motivation we discuss about gapped repeats and palindromes. Gapped repeats and palindromes have been investigated for a long time (see, e.g., [16, 4, 19, 17, 18, 6, 7]), with motivation coming especially from the analysis of DNA and RNA structures, where tandem repeats or hairpin structures play important roles in revealing structural and functional information of the analyzed genetic sequence (see [16, 4, 17]). More precisely, a gapped repeat (respectively, palindrome) occurring in a word w is a factor uvu (respectively, $u^R vu$) of w. The middle part v of such structures is called gap, while the two factors u (or the factors u^R and u) are called left and right arms. Generally, the previous works were interested in finding all the gapped repeats and palindromes, under certain restrictions on the length of the gap or on the relation between the arm of the repeat or palindrome and the gap. In this thesis (and in [9]) we solved the problem in three new different settings.

One should note that the investigation we pursue here is not aimed to tackle real biological facts and provide solutions for them. In fact, its aim is to provide a better understanding of the structural properties of strings obtained by the specific operations discussed as well as specific tools for the manipulation of such strings.

1.2 Thesis structure

The thesis is split into five chapters as follows:

1. The first chapter **"Introduction"** starts by presenting the motivation for the thesis and the place of our results in the research context. In the second part of the chapter personal contributions are presented.

2. In the second chapter **"Preliminaries"** we define most of the notions we use in the thesis. The chapter is split into four sections:

   (a) The first section covers basic definitions about finite and infinite words, factors, periods, runs and finite state machines.

   (b) The second section introduces several operations on strings inspired from DNA biochemistry.

   (c) The third section covers details about the various data structures used in the thesis.

   (d) The fourth and last section presents a few of the lemmas used throughout the thesis.

3. In the third chapter **"Language theoretical and algorithmic aspects"** we present a lot of our main results focusing, as the chapter's title suggests on language theory and algorithmic aspects of our operations. The chapter is split into three sections:

   (a) In the first section we discuss about various duplication languages.

   (b) In the second section we discuss membership problems for the three different operations and also discuss how we can efficiently compute ancestors and common ancestors of words in relation to our operations.

   (c) In the third section we discuss distance related problem in the context of our three operations.

4. The fourth chapter **"Combinatorial aspects"** includes a lot of the important algorithmic and combinatorial results. The chapter is split into three sections:

   (a) The first section tackles the problem of generating finite and infinite words using our operations.

   (b) In the second section we tackle the problem of efficiently detecting the existence of repetitive structures occurring at both ends of some sequence.

(c) In the third section we discuss about gapped repeats and palindromes within a word.

5. In the last chapter **"Future Work"** we present a number of open problems that arise from this thesis.

1.3 Main Results

After discussing about the motivation for our work and pointed out the directions of our research we will now present in detail some of the most important results in the paper.

Though most of the results in the paper are presented in the third and fourth chapter it is worth noting that we feel that the lemmas in section 2.4 are of real interest. Not only because a lot of our results rely on them but also because we feel the technique we used could be useful to prove other results. In section 2.4 we show how we can compute in linear time the length of the minimum or maximum square that starts in each position of the array, or it is centered in each position of the array and other similar results. Although some of the results have already been proved, we bring a different, and from some perspectives, a more general solution. We give a new proof of these results based on a Lempel-Ziv-like factorisation of the input word and a series of combinatorial remarks on the structure of this factorisation. In order to prove these lemmas we also need to use a special case of the disjoint sets union-find problem. Thus, we firstly prove a lemma that shows the disjoint sets union-find problem works in linear time in our problem settings. The results have been published in [10, 9].

In 3.1 we give a few results related to duplication languages that have been published in [12, 13]. We first give a result that clearly separates bounded prefix-suffix duplication languages from prefix-suffix duplication languages:

Theorem 1.3.1 *Every nonempty class of languages closed under union with regular languages, intersection with regular languages, and substitution with regular languages, is closed under bounded prefix-suffix duplication.*

We further prove that we can find if a language is a finite k-prefix-suffix duplication language(that is if it can be generated from a finite set of words through k-prefix-suffix duplication), and also to find such a minimum set if it exists:

Theorem 1.3.2 *Let L be a regular language which is a multiple k-prefix-suffix duplication language for some positive integer k. There exists a unique minimal (with respect to inclusion) regular language E, which can be algorithmically computed, such that $L = PSD_k^*(E)$. In particular, one can algorithmically decide whether L is a finite k-prefix-suffix duplication language.*

In 3.2 we start by solving the membership problem for $PSD_k^*(L)$ languages, provided that we know how to solve the membership problem for L on the RAM with logarithmic word size model.

Theorem 1.3.3 *If the membership problem for the language L can be decided in $\mathcal{O}(f(n))$ time, then the membership problem for $PSD_k^*(L)$ can be decided in $\mathcal{O}(nk\log(k) + n^2 f(n))$.*

Based on the previous result we show that we can decide if a word x is in $PSD_k^*(w)$, with $|x| = n$ in $\mathcal{O}(n\log(k))$ time if $|w| \geq k$ and in $\mathcal{O}(nk\log(k))$ time otherwise. The results above have been published in [12, 13, 10].

We then switch to ancestor (or root) computations and obtained the following result for the prefix-suffix square completion operations:

Theorem 1.3.4 *Given a word w of length n, for the prefix-suffix square completion operation we can identify in $\mathcal{O}(n)$ time:*

1. *The number of ancestors of w*

2. *The shortest ancestor of w*

Based on this result we give an algorithm that verifies the membership problem for this operation in linear time.

Later we show how we can find a common ancestor of two words in $\mathcal{O}(nk\log(k) + n^2)$ time in the case of the bounded prefix-suffix duplication and in linear time for the prefix-suffix square completion. In order to prove the later result we use several tools from linear sorts to suffix arrays and LCP queries, and the lemmas in section 2.4. Using the same tools as before and binary search we obtain an $\mathcal{O}(n\log(n))$ algorithm for finding the shortest common ancestor of two words with respect to the prefix-suffix square completion. Most of the results regarding ancestors appeared in [10, 12, 13, 11] while results concerning common ancestors appear hear for the first time.

In 3.3 we start by giving a $\mathcal{O}(n\log(k))$ time algorithm for computing the bounded prefix duplication distance based on dynamic programming, deques and a series of combinatorial remarks related to runs in a word. Based on the previous result we show how we can compute the bounded prefix-suffix distance between two words:

Theorem 1.3.5 *Given $k \geq 1$, let x and w be two words of respective length m and n, $n > m$. If $m \geq k$, then $\delta_k(x, w)$ can be computed in $\mathcal{O}(n\log(k))$. If $m < k$, then $\delta_k(x, w)$ can be computed in $\mathcal{O}(nk\log(k))$.*

We then give a totally different solution for computing the distance related problems for any of our operations, based on graphs, that has complexity $\mathcal{O}(n^3)$

time. We then prove that we can improve this to $\mathcal{O}(n^2)$ time for the case of the prefix-suffix square completion operation based on some analytical results. We finish the chapter with a result for computing the distance between two bounded prefix-suffix languages:

Theorem 1.3.6 *Given two regular languages L_1 and L_2 over an alphabet V, recognized by deterministic finite automata with sets of states Q and S, respectively, and a positive integer $k \geq 1$, the distance $\delta_k(L_1, L_2)$ can be computed in $\mathcal{O}((k+N)M^2|V|^{2k})$ time, where $M = \max\{|Q|, |S|\}$ and $N = \min\{|Q|, |S|\}$.*

Most of the results in this section have been published in [10, 12, 13].

In 4.1 we show how we can generate different infinite words (like the Fibonacci word and the Thue Morse word) using prefix-suffix square completion. Also we prove that we can't generate the Thue Morse word by prefix-suffix duplications but we can generate Stewart's choral sequence using the PSD operation. Our results are mostly based on thorough analysis of the words and are partly based on computer simulations. The results in this section have been published in [10].

In 4.2 we show how we can construct in linear time a data structure that allows us to verify in constant time if a factor of a word is SD, PD, or PSD primitive. We then give an algorithm that computes all the prefix-suffix-square free factors of a word:

Theorem 1.3.7 *Given a word w of length n, we can find the set S of prefix-suffix-square free factors of w in $\mathcal{O}(n + |S|)$ time.*

We continue by giving an $\mathcal{O}(n \log n)$ algorithm for counting the number of prefix-suffix square free factors that uses segment trees, beside the lemmas in section 2.4. A similar solution is given for finding the longest primitive ancestor of a word in relation to the prefix-suffix square completion operation. We then show that the longest prefix-suffix-square free factor $w[i..j]$ can be computed in linear time, so without enumerating all the elements of S:

Theorem 1.3.8 *Given a word w of length n, we can find its longest prefix-suffix-square free factor in $\mathcal{O}(n)$ time.*

The section ends with a few theorems that provide different factorizations of a word including a factorization in squares proved by this theorem:

Theorem 1.3.9 *Given a word w of length n, we can find (if it exists) in $\mathcal{O}(n \log(n))$ time a factorisation $w = s_1 \cdots s_k$ of w, such that s_i is a square for all $1 \leq i \leq k$.*

Most of the results of this section have been published in [11], but there are also a few results that have been published here for the first time.

In 4.3 we look for factors of the form uvu and $u^r vu$ and compute the longest previous factor ($LPrF$) table, and longest previous reverse factor ($LPrF$) table in three different settings. First, we take into account the case where we want v to be limited by constants both as lower and upper bound. We are able to compute the $LPrF$ table in $\mathcal{O}(n)$ time and the LPF table in $\mathcal{O}(n \log n)$ time in this setting. In order to do so we use a mix of techniques from combinatorics on strings to the union find problem, suffix arrays and we also make use of the dictionary of basic factors. We then compute the two tables in the case where the gap is limited only by a lower bound function. We are able to compute both tables in linear time using similar technique as above but also an interesting technique based on dynamic construction of trees combined with LCP queries. The last case we settle is that of long armed repeats and palindromes. Obtaining solutions of complexity $\mathcal{O}(n + |output|)$ time and $\mathcal{O}(n)$ time. The tool we use in this section are: various existing results on runs and long armed repeats and palindromes in words combined with the lemmas in section 2.4. The results in this section have been published in [9].

Bibliography

[1] Gary Benson. Tandem repeats finder: a program to analyze dna sequences. *Nucleic acids research*, 27(2):573, 1999.

[2] Gary Benson and Lan Dong. Reconstructing the duplication history of a tandem repeat. In *ISMB*, pages 44–53, 1999.

[3] Daniel P. Bovet and Stefano Varricchio. On the regularity of languages on a binary alphabet generated by copying systems. *Information Processing Letters*, 44(3):119 – 123, 1992.

[4] Gerth Stolting Brodal, Rune B. Lyngso, Christian N. S. Pedersen, and Jens Stoye. Finding maximal pairs with bounded gap. In *Proc. 10th Combinatorial Pattern Matching*, volume 1645 of *LNCS*, pages 134–149. Springer, 1999.

[5] S.R. Chan and E.H. Blackburn. Telomeres and telomerase. *Philos. Trans. R. Soc. Lond. B. Biol. Sci.*, 359:109–121, 2004.

[6] Maxime Crochemore, Costas S. Iliopoulos, Marcin Kubica, Wojciech Rytter, and Tomasz Walen. Efficient algorithms for two extensions of LPF table: The power of suffix arrays. In *Proc. SOFSEM 2010*, volume 5901 of *LNCS*, pages 296–307, 2010.

[7] Maxime Crochemore and German Tischler. Computing longest previous non-overlapping factors. *Inf. Process. Lett.*, 111(6):291–295, February 2011.

[8] J. Dassow, V. Mitrana, and Gh. Paun. On the regularity of duplication closure. *Bull. EATCS*, 69:133–136, 1999.

[9] M. Dumitran and F. Manea. Longest gapped repeats and palindromes. In *Proc. MFCS 2015*, LNCS, to appear 2015.

[10] M. Dumitran and F. Manea. Prefix-suffix square completion. In *Proc. WORDS 2015*, LNCS, to appear 2015.

[11] M. Dumitran, F. Manea, and D. Nowotka. On prefix/suffix-square free words. In *Proc. SPIRE 2015*, LNCS, to appear 2015.

[12] Marius Dumitran, Javier Gil, Florin Manea, and Victor Mitrana. Bounded prefix-suffix duplication. In *Proc. CIAA 2014*, volume 8587 of *LNCS*, pages 176–187, 2014.

[13] M. Dumtiran, J. Gil, F. Manea, and V. Mitrana. Bounded prefix-suffix duplication: language theoretic and algorithmic results. *International Journal of Foundations of Computer Science*, to appear.

[14] G. Ehrenfeucht A., Rozenberg. On the separating power of eol systems. *RAIRO - Theoretical Informatics and Applications - Informatique Thorique et Applications*, 17(1):13–22, 1983.

[15] Jesús García-López, Florin Manea, and Victor Mitrana. Prefix-suffix duplication. *J. Comput. Syst. Sci.*, 80(7):1254–1265, 2014.

[16] Dan Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, USA, 1997.

[17] Roman Kolpakov and Gregory Kucherov. Searching for gapped palindromes. *Theor. Comput. Sci.*, 410(51):5365–5373, 2009.

[18] Roman Kolpakov, Mikhail Podolskiy, Mikhail Posypkin, and Nickolay Khrapov. Searching of gapped repeats and subrepetitions in a word. In *Proc. CPM*, volume 8486 of *LNCS*, pages 212–221, 2014.

[19] Roman M. Kolpakov and Gregory Kucherov. Finding repeats with fixed gap. In *Proc. SPIRE*, pages 162–168, 2000.

[20] P. Leupold, J. Karhumäki, V. Mitrana, Universitat Rovira i Virgili. Facultat de Lletres de Tarragona, and Universitat Rovira i Virgili. Departament de Filologies Romàniques. *Languages Generated by Iterated Idempotencies: PhD Dissertation : Tesi Doctoral*. Universitat Rovira i Virgili, 2008.

[21] Peter Leupold, Victor Mitrana, and JosM. Sempere. Formal languages arising from gene repeated duplication. In Nataa Jonoska, Gheorghe Pun, and Grzegorz Rozenberg, editors, *Aspects of Molecular Computing*, volume 2950 of *Lecture Notes in Computer Science*, pages 297–308. Springer Berlin Heidelberg, 2004.

[22] M. Lothaire. *Combinatorics on Words*. Cambridge University Press, 1997.

[23] J.P. Murnane. Telomere dysfunction and chromosome instability. *Mutat. Res.*, 730:28–36, 2012.

[24] R.J. Preston. Telomeres, telomerase and chromosome stability. *Radiat. Res.*, 147:529–534, 1997.

[25] Dina Sokol, Gary Benson, and Justin Tojeira. Tandem repeats over the edit distance. *Bioinformatics*, 23(2):e30–e35, 2007.

[26] L. D. Stein. Human genome: end of the beginning. *Nature*, 431(7011):915–916, October 2004.

[27] I. Wapinski, A. Pfeffer, N. Friedman, and A. Regev. Natural history and evolutionary principles of gene duplication in fungi. *Nature*, 449:54–61, 2007.

[28] Ming wei Wang. On the irregularity of the duplication closure. *Bulletin of the EATCS*, 70:162–163, 2000.