On the Multiset of Factors of a String

A thesis submitted in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

Himadri Nayak
(Roll Number: 09612315)

to the

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
Guwahati 781039, India

January, 2017

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

DECLARATION

It is certified that the work contained in the thesis titled **"On the Multiset of Factors of a String"** has been done by me, a student in the Department of Mathematics, Indian Institute of Technology Guwahati under the guidance of **Dr. Kalpesh Kapoor** for the award of Doctor of Philosophy and that this work has not been submitted elsewhere for a degree.

January, 2017

Himadri Nayak
Department of Mathematics
Indian Institute of Technology Guwahati

CERTIFICATE

It is certified that the work contained in the thesis titled **"On the Multiset of Factors of a String"** by **Himadri Nayak**, a student in the Department of Mathematics, Indian Institute of Technology Guwahati for the award of the degree of Doctor of Philosophy has been carried out under my supervision and this work has not been submitted elsewhere for a degree.

January, 2017

Dr. Kalpesh Kapoor
Associate Professor
Department of Mathematics
Indian Institute of Technology Guwahati

Abstract

A string is a finite or infinite sequence of letters drawn from an alphabet. A string v is said to be factor of another string s if there exists strings u and w such that $s = uvw$. of This thesis investigates properties of strings that have the same multiset of factors of certain length. Two strings u and v are called k-abelian equivalent if the frequency of every factor of length $\leq k$ are the same in u and v. Alternatively, the set of strings with identical multiset of factors of length up to k are said to k-abelian equivalent. The existing characterization of the permutations that are required to transform a string to a different string having the same multiset of factors of length k are used for showing that it is sufficient to check k and $(k-1)$-length factor multiset of two strings to identify if they are k-abelian equivalent. A necessary and sufficient condition has been given for two strings to be k-abelian equivalent. The number of strings having identical multiset of k-length factors can be computed using the existing technique for finding the number of Eulerian cycles in a multidigraph. We use a variation of this theory to find a lower bound for the maximum number k-abelian equivalent strings for $k = 2$. Also the maximum number of strings of length n that are $\lfloor \frac{n}{2} \rfloor$-abelian equivalent is given. A generalization of k-abelian equivalence of words, k-precedence data equivalence, that takes into account the order of appearance of two same length factors is investigated. We study the related permutation that are required to transform one string to the other having the same k-precedence data. It is shown that the strings with length n can be transformed under some specific rules to a different string with the same k-precedence data if and only if $k \geq \lfloor \frac{n-1}{3} \rfloor + 1$. We refine the definition of l-modular factor composition of words, which is a generalization of composition of a word given by the multiset of factors of certain length. A relation with k-abelian equivalence of words in this context is established. We introduce l-modular factor precedence data as a generalization of both l-modular composition and l-precedence data of a word. A lower bound of $\Omega\left(\sqrt[4]{\frac{n}{\log n}}\right)$ on l is given to have different l-modular factor precedence data for every n-length string. We reduce the problem of finding non zero discrepancy with respect to partial coloring on a particular kind of l-regular hypergraph to the problem of finding different l-modular compositions, a restricted version of l-modular factor

composition, in a pair of strings. Then the existing bounds on l to generate different l-modular composition for every n length string is used for obtaining the bounds on l to have non zero discrepancy in l-regular hypergraph with respect to a partial coloring.

Acknowledgement

I would like to express my gratitude to my thesis supervisor Dr. Kalpesh Kapoor, Associate Professor, Department of mathematics, IIT Guwahati. Not only his guidance but also his patience and support over the years helped me to complete writing this thesis. I cannot thank him enough for introducing me to various research topics and also for giving repeated emphasis on the ability to form research questions. This thesis has been shaped up in many ways by his organised attitude and work ethics.

I am very thankful to my doctoral committee members Dr. Partha Sarathi Mandal, Prof. Prabin Kumar Bora and Dr. Pinaki Mitra for their advice and valuable suggestions.

I would like to take this opportunity to thank Academic Section, Financial and Accounts Section of IIT Guwahati and University Grants Commission for taking care of my scholarship related matters. A special thanks would go in this regard to Prof. S.N. Bora and Prof. Bhaba Kumar Sharma, the present and former HOD's of Department of Mathematics, IIT Guwahati and also Dr. S.K. Barua, Assistant Registrar of IIT Guwahati.

I would like to thank Dr. Sutanu Roy, both as a dear friend and the person who inspired me to do research in Mathematics. As an academician he has been truly helpful to me and I would like to take this opportunity to thank him for that.

I am thankful to my fellow researchers Shib Shankar Das, Swarup Panda, Dr. Biswajit Deb, Ananda Nayak for their valuable inputs and for the discussions with them that has helped me in many ways.

I will forever be thankful to Barun Gorain, Dr. Santu Das, Dr. Kaushik mondal, Kalyan Manna, Arnab Ghosh, Debopam Chakraborty, Dr. Murali Reddy, Abhishek Das, Anirban Majumdar, Projesh Nath Choudhury, Dishari Chowdhury and also Niladri Sett, Purnendu Mondal, Dr. Avijit Ghosh, Dr. Debjyoti Sahu, Shuvendu Rana, Pramit Nandi for being a part of my journey in IIT Guwahati. All these years they were with me in the ups and downs of my life like good family members do. A special thanks goes to Mandar Maitra for the creative motivations and also for being the creative inspiration himself and above all for being the true friend that he is.

I would like to thank Dr. K.V. Srikanth, Dr. Anjan Kumar Chakraborty, Prof. Sukanta Pati. Their classes were inspirational and their impacts have been multi-faceted in a positive way.

I shall be eternally grateful to the facilities provided by IIT Guwahati and more specifically to the Department of Mathematics. I would like to take this opportunity to thank Mr. Santanu Majumdar, Mr. Sridhar Samal, Mr. Phatik Kumar, Mr. Pranpratim Borgohain for their help over the years.

At last but not the least I am grateful to my family, my father, my mother, my little brother and my grand mother. Without their loving support I could not have pursued higher studies. Their little sacrifices, values and most importantly unconditional affection and love has enriched me as a person.

January, 2017 **(Himadri Nayak)**

List of Symbols

Σ	An alphabet, a finite set of symbols	1
Σ^n	Set of all n-length strings whose letters are from the alphabet Σ	7
$u \cdot v$ or uv	Concatenation of two strings u and v	7
u^l	$u^2 = u \cdot u$ and $u^l = u \cdot u^{l-1} = u^{l-1} \cdot u$	7
$\|u\|$	Length of the string u if u is a string OR Cardinality of the set u if u is a set	7
$\|u\|_v$	Frequency of a string v in the string u as a factor	7
$\text{pref}_k(u)$	Prefix of length k of the string u	7
$\text{suff}_k(u)$	Suffix of length k of the string u	7
$m_k(u)$	Multiset of k-length factors of the string u	7
$m_{[1,k]}(u)$	Multiset of factors of length up to k of the string u	8
$\mathbb{M}_k(u)$	Set of all strings v such that $m_k(v) = m_k(u)$	7
$\mathbb{M}_{[1,k]}(u)$	Set of all strings v such that $m_{[1,k]}(v) = m_{[1,k]}(u)$	8
$T_k(u)$	k-tuple notation of the string u	8
G_k	De Bruijn graph of order k	20
G_u^k	Rauzy multigraph of the string u with respect to k	22

List of Symbols

F_k^s	Lexicographically ordered set of k-length factors of the string s	40
OP_k^s	A square matrix where the entry $OP_k^s(i,j)$ is the number of ways $F_k^s(i)$ and $F_k^s(j)$ can appear as factors in s at positions with indices γ and η respectively where $\gamma < \eta$	41
$OP_k(u)$	Multiset of ordered pairs of k-length factors of the string u; It can be expressed by the ordered pair (OP_k^s, F_k^s)	40
$\lvert s\rvert_{(w,v)}$ or $OP_k^s(w,v)$	If $F_k^s(i) = w$ and $F_k^s(j) = v$ then $\lvert s\rvert_{w,v} = OP_k^s(w,v) = OP_k^s(i,j)$	41
${}_a^d F_k^s$	The lexicographically ordered set of all k length factors of s, which appear in s at positions congruent to a mod d	64
$[{}_a^d \mathcal{M}_k^s]$	A vector where $[{}_a^d \mathcal{M}_k^s](i)$ is the frequency of the string ${}_a^d F_k^s(i)$ in the string s at position congruent to a mod d	64
$[{}_{(a,b)}^{\ d}\mathcal{OP}_k^s]$	A matrix where the entry $[{}_{(a,b)}^{\ d}\mathcal{OP}_k^s](i,j)$ is the number of ways the string ${}_a^d F_k^s(i)$ appears at positions congruent to a mod d before the string ${}_b^d F_k^s(j)$ at positions congruent to b mod d as factors in the string s	69

List of Figures

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Contents

Chapter 1

Words and its Factors: An Introduction

Let Σ be a finite set of symbols. By a *"word"* or *"string"* we mean a sequence of symbols drawn from Σ. Here the sequence may be finite or infinite. A finite subsequence of a word where the indices of the subsequence are consecutive is called a *substring* or *factor*. Combinatorics on words is an extensively studied topic and there are several books available on different aspects of finite and infinite word combinatorics, see for example [5, 6, 18, 31–33]. We focus on finite words and this thesis is concerned with the investigation of properties various compositions of a string based on its multisets of factors.

One of the focus of combinatorics on words is the study of the relation of words and its factors. Ilie and Smyth [25] have studied minimum unique factors of a string and explores its relation with most repeated factors. They have given algorithms to compute one from the another. An algorithm to efficiently compute a shortest word which is not a factor of a given word is given in [58]. Such words are referred to as minimal absent words and have applications in several areas of bioinformatics such as in preventive and curative medical applications where they can be used as biomarkers [1, 19, 21].

All these problems are related to some composition of a word given by its factors. Finding some unique property in a string from a collection of strings from some

composition based on their factors is related to data compression and complexity of strings. Complexity of a word is a measure how much space is necessary to describe the string uniquely. Kolmogorov-Chitin complexity of words [30] is the size of the shortest pair including a Turing machine description and an input such that the Turing Machine gives output the given string on the input string. The alternative definitions of word complexity are also explored which relates a word with various models of computation. Shallit and Wang [52] explores the automatic complexity of a word $x \in \Sigma^n$ which is defined as the smallest number of states in any deterministic finite automata M such that the set of words of length n that are accepted by M is equal to the set $\{x\}$.

Factor complexity of an infinite word is the number of different n-length factors of the infinite word. Two words are called k-abelian equivalent if and only if multisets of factors of length up to k is the same for those two words. The number of equivalence classes formed by this equivalence relation on the n length factors of an infinite string is the k-abelian complexity [29] of that infinite word.

Generating different k-abelian equivalent words of any string is computing all possible reconstructions from the given string's multiset of factors of length up to k. Now, reconstruction of a string from the multiset of factors of length k is a well studied problem that has its origin in bioinformatics. A similar problem is to reconstruct a genome sequence from short fragments. The process of physical mapping of a DNA actually starts with breaking into small pieces. One such problem is Shortest Superstring Problem. This problem also has application in the area of data compression [55]. The problem of computating shortest superstring for a set of strings is known to belong to NP-Complete complexity class even when restricted to binary alphabet [15]. A special case of shortest superstring problem is Sequencing by Hybridization (SBH) [4, 11, 34, 53] in which the lengths of the factors in the input multiset are identical. Ukkonen [56] introduced this problem. Difference between the multisets of factors of a fixed length of two strings is then used to define the notion of distance between the strings. In [56], a set of transformation rules were proposed and it was conjectured that they can used to generate the set of words that have the same multiset of factors. The conjecture was proved later in [39] using graph

theoretic approach.

If a string u of length n is given along with its multiset of l-length factors then the reconstructed strings may not be the same as u. In that case it is called an ambiguous reconstruction of u from its multiset of l-length factors.

The number of reconstructions (the original string and ambiguous ones) of u is given by the number of Eulerian paths (or cycles) in the directed multigraph whose construction is similar to that of the De Bruijn Graph [9, 16]. We call this multigraph a Rauzy multigraph and is defined in Chapter 3. Here, unlike in the case of De Bruijin graph where the nodes are all possible strings of a fixed length, the nodes are all possible $(l-1)$-length factors of the string under consideration and the edges correspond to the multiset of l-length factors. This approach has been discussed in [39].

The formula that gives the number of Eulerian cycles in a digraph is known as BEST theorem [14]. The number of Eulerian paths in a digraph can be obtained with a minor modification in the formula in BEST theorem.

The SBH problem is known to be solvable in polynomial time [38] [28, Chapter 2]. But the ideal case of SBH demands the information about l-tuple composition of a string or the multiset of all l-length factors of the DNA string to be known precisely. In a real life situation we may not get the exact l-tuple composition. Usually while reading DNA sequences, due to mechanical and chemical errors, either some less information about the multiset is obtained or some erroneous extra information is received. The combinatorial problem to reconstruct a string from such type of information on l-tuple composition has been proved to be NP complete [3].

To resolve the problem of ambiguity of a solution in SBH, alternatives have been investigated, that attempts to specify position of a factor as part of input. One such attempt is Sequencing by Nested Strand Hybridization(SNSH) which has been developed in [7, 43]. This procedure uses not only multiset of factors of fixed length of a word but also the information of their pairwise order or precedence data. Rubinov and Gelfand [49] investigated a version of this problem where the aim is to reconstruct a single string. But their reconstruction did not take multiplicity of order information into consideration and also reconstruction length is arbitrary.

Availability of positions of each k-length factors in the multiset of k-length factors of a string leads to Positional Sequencing By Hybridization (PSBH) [20]. In SBH problem the information available about a string is factors of a certain length and their frequencies. Problems such as SNSH, PSBH add some more information to the factors of the same length. Another way to have additional information to the multiset factors of a certain length is having multiset of factors of all the lengths up to that length. This idea was discussed in [29] and [40]. They have independently came to few common conclusions.

Every reconstruction process separates strings from all other strings up to ambiguity. A dual of this problem is to find unique characteristic of a string that distinguishes it from given other strings. If two strings are given the problem of distinguishing them with the help of various characteristics of the strings is known as Separating Words problem. One such problem is to find automatic complexity where the target set is a singleton set of a string [52].

The problem of separating words by a Deterministic Finite Automata (DFA) has been introduced by Goralčík and Koubek [17]. They asked the question: For two arbitrary strings, what is the minimum number of states required in a DFA which will accept one and reject the other string? They have proved an $O(\log n)$ bound for any pair of strings with maximum length n. They also investigated the minimum number of states needed for DFA's separating any pair of strings of length at most n. They came up with an $O(n)$ bound. Later Robson [46, 47] reduced the bound to $O(n^{\frac{2}{5}} \log^{\frac{3}{5}} n)$. In [47] for a particular class of DFA's, the bound was proved to be $O(\sqrt{n})$.

1.1 Organization of the Thesis

The following problems are investigated in this thesis.

Problem 1.1. Can we find a pair of strings u and v whose multiset of k-length and $(k-1)$-length factors are the same and u is not k-abelian equivalent to v?

The problem was given as an open question in [40]. A related result that we

prove is about the nature of suffix and prefix of k-abelian equivalent words using the idea of Eulerian paths [39]. We also find the relation between the cardinality of the set of words which are k-abelian equivalent and the cardinality of the set of words where they all have the same multiset of k-length factors. The former cardinality gives a measure of the complexity of a word from that set. It is natural to ask how much complex a word can get in this regard. As a first step, we have solved the following problem:

Problem 1.2. What is the maximum cardinality of the set of k-equivalent n-length words where $k = \frac{\lfloor n \rfloor}{2}$ and $k = 2$?

Consider the set of ordered pairs of a factors of a word and give that set a miltiset structure by combining frequencies to each member of the set. The frequency comes from the *number of ways* the first member of the ordered pair appears first and then the second member appears in the word as factors if we index the letters of the word from left to right. We call this multiset k-factor precedence data if the factors in this context are all of length k. It leads to a generalization of k-abelian equivalence. So, it is natural to ask the following question for which we give heuristics. Polynomial time algorithm for this problem is not known yet.

Problem 1.3. How can we reconstruct a string from its k-precedence data?

Two different words can have the same k-precedence data. So the process of reconstruction of a word from its k-precedence data can result in a different word. In most of the cases we have more than one different reconstructions of a word. We partially solve the following problem

Problem 1.4. What is the structure and minimum length of the string pairs having the same k-precedence data?

We found the structure and minimum length of the string pairs which are obtained by applying a specific kind of transformation. This transformation keeps the k-precedence data on a string intact. If we define a relation on the words such that two words are related if and only if they are obtained from each other by using

this transformation. It is yet unknown whether a set of all strings having the same k-precedence data is different from the transitive closure of the relation mentioned above. Also we checked experimentally that for the value of k up to 10, there is no difference between the answers to the Problem 1.4 and the the problem we have solved in this regard.

We investigate the relation between partial coloring of hypergraph and modular composition of words which was introduced by [57] in 2014. We give a tight bound for a special class of hypergraphs.

Problem 1.5. What is the lower and upper bound of discrepancy of a hypergraph with respect to a partial coloring?

In, [57] modular composition of a word has been generalised and used as a tool for making finite automata for separating words. We have further generalised the idea by using arithmetic progression and precedence data on k-tuple notation of a word and found few results on separating word problem.

The rest of the chapters in this thesis are organised as follows:

- **Chapter 2:** We discuss the k-abelian equivalence and the solution to Problem 1.1.
- **Chapter 3:** We present the relation between k-abelian equivalence and Eulerian path and their enumeration and solution to Problem 1.2 is given.
- **Chapter 4:** We introduce the idea of k-factor precedence data of a string and relate it to k-abelian equivalence. We discuss the difficulties on reconstructions from k-precence data of a string. We present a solution to Problem 1.4 for a particular class of reconstruction.
- **Chapter 5:** We discuss separating words using multisets of factors and the Deterministic Finite Automata (DFA) that can be created using the properties of factors. We relate it to discrepancy theory and solve Problem 1.5.
- **Chapter 6:** We draw conclusions from our work and then present few related open problems.

Chapter 2

k-Abelian Equivalent Words

2.1 Multiset of Factors and k-transformations

Let Σ be a finite alphabet. A string or a word is a finite sequence of letters chosen from an alphabet Σ. The length of a string u is denoted by $|u|$. The string with length 0 is referred to as ϵ or empty. The set of all finite length strings including ϵ is denoted by Σ^*. The set of all strings of length n formed by the letters from the alphabet Σ is denoted by Σ^n.

If two finite strings u and v are put one after another like vu the $s = vu$ is again a finite string. s is called the concatenation of the strings v and u respectively. Sometimes s is written as $v \cdot u$ also. If $u = v$ then s is written as $S = uu = u^2$. Similarly we say $s = u^l$ if u is juxtaposed with u $l-1$ times. A string x is said to be a factor of another string y if and only if $y = pxs$, where p and s are some strings in Σ^*. If $p = \epsilon$ then x is said to be a prefix of y. Similarly, if $s = \epsilon$ then x is said to be a suffix of y. A prefix and suffix of length l of string y ($0 \leq l \leq |y|$) are denoted by $\text{pref}_l(y)$ and $\text{suff}_l(y)$, respectively. We define $\text{pref}_0(y) = \text{suff}_0(y) = \epsilon$.

Let u and x be two non empty words. The number of times x appears as a factor in u is denoted by $|u|_x$. For example, $|1010101|_{101} = 3$. We define a relation $\overset{k}{\equiv}$ as follows: $u \overset{k}{\equiv} v$ if and only if $m_k(u) = m_k(v)$, where $m_k(u)$ and $m_k(v)$ are multisets of k-length factors of u and v, respectively. This is an equivalence relation and let us call it k-factor equivalence relation. We define $\mathbb{M}_k(u)$ as the set of strings

whose k-length factor multisets are equal to $m_k(u)$. Based on this relation we define another equivalence relation $\overset{k}{\underset{r}{\equiv}}$ as follows: $u \overset{k}{\underset{r}{\equiv}} v$ if and only $m_i(u) = m_i(v)\ \forall i \in \{r, r+1, \ldots, k-1, k\}$. We say $\mathbb{M}_{[r,k]}(u) = \{v | u \overset{k}{\underset{r}{\equiv}} v\}$ and $m_{[r,k]}(u) = \bigcup_{r \le i \le k} m_i(u)$. The special case where $r = 1$ the words u and v with the property $u \overset{k}{\underset{1}{\equiv}} v$ are called k-abelian equivalent. The k-abelian equivalence has been introduced in [29]. The main result of this chapter has been proved later in [29] independently.

Note that $m_k(u) = m_k(v) \Leftrightarrow u \overset{k}{\equiv} v \Leftrightarrow \mathbb{M}_k(u) = \mathbb{M}_k(v)$.

When $m_k(u) = m_k(v)$ for strings u and v, we have k-length factors of u changing order in v. We can think of writing k-length factors of u sequentially as they occur in u and then have a permutation on the sequence of k-length factors and merge them to get v. Let us define k-tuple notation of a string.

Definition 2.1 (k-tuple notation of a string)**.** Let $u = u_1 u_2 \ldots u_n$ be a string of length n where $u_i \in \Sigma\ \forall i \in \{1, \ldots, n\}$ and Σ is an alphabet. We define $T_k(u)$ to be the k-tuple notation of u as the string $t_1 t_2 \ldots t_l$ where $t_i \in \Sigma\ \forall i \in \{1, \ldots, l\}$ and $l = (n-k+1)k$ and $t_{(i-1)k+1} t_{(i-1)k+2} \ldots t_{ik} = u_i u_{(i+1)} \ldots u_{(i+k-1)} \forall i \in \{1, \ldots, n-k+1\}$.

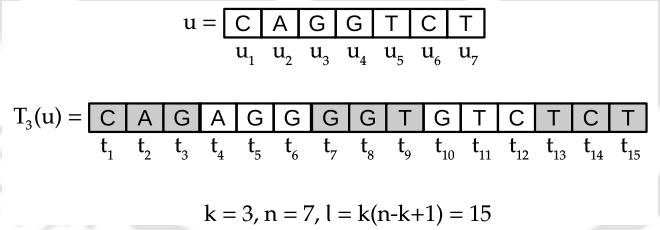

Figure 2.1: The 3-tuple notation of string u represented by $T_3(u)$.

When a string v is reconstructed from $m_k(u)$ then some rules are required to be followed. Ukkonen [56] conjectured that u and v always can be transformed to each other by some Transpositions and Rotations when they are written in $(k-1)$-tuple notation. Pevzner [39] proved this conjecture. We present the definitions of those transformations here:

Definition 2.2 (k-transposition [56])**.** Let y and y' be two strings that can be expressed as either of these two forms given below.

$$(1^{st}\ form)\begin{bmatrix} T_{(k-1)}(y) & = y_1z_1y_2z_2y_3z_1y_4z_2y_5 \\ T_{(k-1)}(y') & = y_1z_1y_4z_2y_3z_1y_2z_2y_5 \end{bmatrix}$$

or

$$(2^{nd}form)\begin{bmatrix} T_{(k-1)}(y) & = y_1zy_2zy_3zy_4 \\ T_{(k-1)}(y') & = y_1zy_3zy_2zy_4 \end{bmatrix}$$

where $|z_1| = |z_2| = |z| = k-1$ and $y_i \in \Sigma^*$ and $|y_i| = j_i(k-1)$ for some non negative integers j_i, $\forall i \in \{1,2,3,4,5\}$. Then the two strings y and y' are k-transpositions of each other.

Definition 2.3 (k-rotation [56])**.** Let y and y' be two strings that can be expressed as $T_{(k-1)}(y) = z_1y_1z_2y_2z_1$ and $T_{(k-1)}(y') = z_2y_2z_1y_1z_2$ where, $|z_1| = |z_2| = k-1$ and $y_1, y_2 \in \Sigma^*$ and $|y_1| = j_1(k-1), |y_2| = j_2(k-1)$ for some non negative integers j_1 and j_2. Then the two strings y and y' are k-rotations of each other.

As these transformations preserve k-length multiset of strings we shall call them k-transformations. We will denote the first and second form of transpositions between two strings y and y' by k-tr_1 and k-tr_2, respectively. Similarly, we will refer to rotations with $z_1 \neq z_2$ and $z_1 = z_2$ by k-rt_1 and k-rt_2, respectively.

Lemma 2.4 (The Ukkonen's conjecture [56] (Proved by pevzner) [39])**.** Let u and v be two distinct strings such that $u \overset{k}{\equiv} v$. Then, u and v can be transformed into each other by application of one or more k-transpositions and k-rotations.

2.2 Effects Due to k-transformations

Any permutation on a multiset can be expressed as effect of applying several basic permutations on the multiset. In particular every permutation can be written as product of transpositions.

If $v \in \mathbb{M}_k(u)$ then $T_k(v)$ is also a permutation on $T_k(u)$. But any arbitrary permutation on $T_k(u)$ resulting the string s does not guarantee existence of a v such that $T_k(v) = s$. We have this constraint because while incorporating such special type of permutation, a k-tuple cannot be placed arbitrarily unless the the next (if

at all exists) or previous (if at all exists) k-tuple forms a $(k-1)$-overlap with the k-tuple in consideration.

Lemma 2.4 proves that in spite of the constraint, one can express the permutation involved in transforming $T_k(u)$ to $T_k(v)$ as the effect of applying several special and basic permutations namely k-transposition and k-rotations defined in Definition 2.2 and Definition 2.3 respectively.

We denote the action of a k-rt_1 rotation by $y \xrightarrow[k\text{-}rt_1]{zz'} y'$ where y and y' are strings as given in Definition 2.3. Let u, v and w be three strings. A non-empty string, s, is said to *spread* over a string uv if $u = u_1u_2, v = v_1v_2$ and $s = u_2v_1$, where $|u_2|, |v_1| > 0$. Similarly, s spreads over a string uvw if $u = u_1u_2, w = w_1w_2, s = u_2vw_1$ and both u_2, w_1 are non-empty strings.

Lemma 2.5. Let $u \overset{k}{\equiv} u'$ such that u' is obtained from u by applying any one of the k-tr_1, k-tr_2 or k-rt_2 transformation. Then,

(a) $\text{pref}_{k-1}(T_{k-1}(u)) = \text{pref}_{k-1}(T_{k-1}(u'))$ and $\text{suff}_{k-1}(T_{k-1}(u)) = \text{suff}_{k-1}(T_{k-1}(u'))$.

(b) $T_{k-1}(u) \overset{i}{\equiv} T_{k-1}(u')\ \forall i \leq k$.

Proof.

(a) This is straightforward from the Definitions 2.2 and 2.3. Observe that the prefix y_1z_1 or y_1z and the suffix z_2y_5 or zy_4 of lengths at least $k-1$ in the k-transposition remains the same. Also prefix (suffix) of u of length at least $k-1$ is the same in $T_{(k-1)}(u)$. Similarly, both the suffix and prefix of length $k-1$ in a k-rt_2 rotation does not change.

(b) Let s be any string of length l, where $l \leq k-1$. Assume that the strings u and u' are of the form as given in either Definition 2.2 or 2.3. We consider the following three cases.

(i) s is a factor of any of the following strings: y_1, y_2, y_3, y_4, y_5, z_1, z_2, z_3 or z.

(ii) s is spread over any of the following strings:
$y_1z_1, y_2z_2, y_3z_1, y_4z_2, z_1y_2, z_2y_3, z_1y_4, z_2y_5$ (in case of k-tr_1 transformation);

$y_1z, y_2z, y_3z, zy_2, zy_3, zy_4$ (in case of k-tr_2 transformation);
zy_1, zy_2, y_1z, y_2z (in case of k-rt_2 transformation, where $z_1 = z_2 = z$).

(iii) s is spread over any of the following strings:
$z_1y_2z_2, z_2y_3z_1, z_1y_4z_2$ (in case of k-tr_1 transformation);
zy_2z, zy_3z (in case of k-tr_2 transformation);
zy_1z, zy_2z (in case of k-rt_2 transformation, where $z_1 = z_2 = z$).

Apart from the above three cases s cannot be a factor of $T_{k-1}(u)$ or $T_{k-1}(u')$ in any other possible way. For example, s cannot be spread over a segment $y_1z_1y_2$ as $|z_1| = k-1$. We note that in all the above three cases the segments are only changing their positions in u' from u. Which means that the number of times the segments appear in u and u' is the same. Thus, their frequencies remain the same. This means that there cannot be any string s such that $|s| < k$ and $|T_{k-1}(u)|_s \neq |T_{k-1}(u')|_s$ if $u \overset{k}{\equiv} u'$.

□

To elaborate the last part of the above proof, let us consider an example where the string X can be written as VW where V and W are also strings. Let $\{w_1, w_2, \ldots, w_n\}$ is the set of all the factors of W and $\{v_1, v_2, \ldots, v_m\}$ is the set of all the factors of V. Also let $Y = VW$. Then $|X|_{w_i} = |Y|_{w_i}$ and $|X|_{v_j} = |Y|_{v_j}$ if w_i and v_j do not appear as factors in the junction of W and V for both x and Y. for all $i \in \{1, 2, \ldots, m\}$ and $j \in \{1, 2, \ldots, n\}$. But if somehow the junctions also appear the same number of times in X and Y, then the condition can be withdrawn.

Corollary 2.6. Let $u \overset{k}{\equiv} u'$ such that u' is obtained from u by applying any one of the k-tr_1, k-tr_2 or k-rt_2 transformation. Then,

(a) $\text{pref}_{k-1}(u) = \text{pref}_{k-1}(u')$ and $\text{suff}_{k-1}(u) = \text{suff}_{k-1}(u')$.

(b) $u \overset{i}{\equiv} u'$ $\forall i \leq k$.

Proof. As $\text{suff}_{k-1}(u) = \text{suff}_{k-1}(T_k(u))$ for any string u, the first result follows directly from the first result of Lemma 2.5.

The statement (b) can be proved in the same way as the statement (b) in Lemma 2.5 as any factor of u of size $\leq (k-1)$ will be in the segments y_i or z_i and not going to be spread over the junctions of those segments in $T_{k-1}(u)$. In this way the frequency of that factor also remains the same in $T_{k-1}(u')$ and subsequently in u'. □

The above results show k-transformations except $k - rt_1$-rotation produces k-abelian equivalent strings. If our goal is to keep the k-abelian equivalence after repeated use of k-transformations to a string, then one may ask that if at all $k-rt_1$-rotation can ever be applied. The next result tells us that although applying only once, $k - rt_1$-rotation may not keep the k-abelian equivalence, the change in the multiset of factors up to length $k-1$ of the resultant string is confined to the suffix and prefix of the two strings.

Lemma 2.7. Let $u \xrightarrow[k\text{-}rt_1]{zz'} u'$ where $T_k(u)$ and $T_k(u')$ are of the form $zyz'y'z$ and $z'y'zyz'$, respectively, where $|z| = |z'| = k-1$. Let s be a string of length less than k. Then, $|T_{k-1}(u)|_s - |T_{k-1}(u')|_s = |z|_s - |z'|_s = |u|_s - |u'|_s$.

Proof. Let $|z|_s = \lambda$ and $|z'|_s = \gamma$ $(\neq \lambda)$. In addition to s being a factor of segments y, y', z and z', s can spread over the segments $zy, yz', z'y', y'z, zyz'$ and $z'y'z$ of u. However, s cannot spread over the segment $yz'y'$ (and $y'zy$) as $|s| \leq |z'| = k-1$. All these segments (over which s can spread) are also present in $T_k(u')$ with the identical frequency. If at all s is present anywhere else other than the strings z and z', the application of k-transformation will only change their positions without changing its frequency. Let the frequency of such s that do not occur in either z or z' is δ and the frequency of s that occur at y and y' only is δ'. This implies

$$
\begin{aligned}
|T_{k-1}(u)|_s &= 2\lambda + \gamma + \delta, \\
|T_{k-1}(u')|_s &= \lambda + 2\gamma + \delta, \\
|u|_s &= 2\lambda + \gamma + \delta' \text{ and} \\
|u'|_s &= \lambda + 2\gamma + \delta'.
\end{aligned}
$$

Thus, $|u|_s - |u'|_s = |T_{k-1}(u)|_s - |T_{k-1}(u')|_s = \lambda - \gamma = |z|_s - |z'|_s$.

□

The next result shows that even multiple application of the $k - rt_1$ rotation also confines the changes in the multiset of factors up to length $k-1$ in the suffix and prefixes of the starting and the resultant string.

Lemma 2.8. Let $u_i \xrightarrow[k\text{-}rt_1]{z_i z_{i+1}} u_{i+1}$ $\forall i \in \{1, 2, \ldots, d-1\}$, $d > 1$. Further s be any string of size less than k. Then, $|T_{k-1}(u_1)|_s - |T_{k-1}(u_d)|_s = |u_1|_s - |u_d|_s = |\text{pref}_{k-1}(u_1)|_s - |\text{pref}_{k-1}(u_d)|_s$.

Proof. We shall use Lemma 2.7 and apply mathematical induction on d.

From Lemma 2.7, we know that the statement is true for $d = 2$. Let the statement be true for some $n(\geq 2)$. Equivalently, if $u_i \xrightarrow[k\text{-}rt_1]{z_i z_{i+1}} u_{i+1}$ $\forall i \in \{1, 2, \ldots, n-1\}$ and s be any string of size less than k then:

$$|u_1|_s - |u_n|_s = |\text{pref}_{k-1}(u_1)|_s - |\text{pref}_{k-1}(u_n)|_s \tag{2.1}$$

Assume that there exists u_{n+1} such that $u_n \xrightarrow[k\text{-}rt_1]{z_n z_{n+1}} u_{n+1}$ where $|z_n| = |z_{n+1}| = k-1$. From Lemma 2.7, we have:

$$|u_n|_s - |u_{n+1}|_s = |\text{pref}_{k-1}(u_n)|_s - |\text{pref}_{k-1}(u_{n+1})|_s \tag{2.2}$$

Adding both sides of the Equations (2.1) and (2.2) we get:

$$|u_1|_s - |u_{n+1}|_s = |\text{pref}_{k-1}(u_1)|_s - |\text{pref}_{k-1}(u_{n+1})|_s \tag{2.3}$$

Thus the statement of this lemma is true for any value of $d \geq 2$.

Here k-rt_1 rotations are applied to each of $u_1, u_2, \ldots, u_{d-1}$. So, those u_i's are of the form $z_i y_i z_i' y_i' z_i$, where $|z_i| = |z_i'| = k-1$ $\forall i \in \{1, 2, \ldots, d-1\}$. Let, $|z_i|_s = \lambda_i, |z_2'| = \gamma_i$ $\forall i \in \{1, 2, \ldots, d-1\}$. Also, for each u_i where $i \in \{1, 2, \ldots, d-1\}$, by δ_i we express the frequency of occurrences of s in u_i barring those where s appears in z_i or z_i'. Here we also see that $\forall i \in \{2, 3, \ldots, d\}$ after applying k-rt_1 rotations u_i's are of the form $z_{i-1}' y_{i-1}' z_{i-1} y_{i-1} z_{i-1}'$. Hence, the frequencies of s in each u_i can be expressed as follows:

$$\begin{aligned}
|u_1|_s &= 2\lambda_1 + \gamma_1 + \delta_1 \\
|u_2|_s &= 2\lambda_2 + \gamma_2 + \delta_2 &&= 2\gamma_1 + \lambda_1 + \delta_1 \\
|u_3|_s &= 2\lambda_3 + \gamma_3 + \delta_3 &&= 2\gamma_2 + \lambda_2 + \delta_2 \\
\vdots &\quad \vdots &&\quad \vdots \\
|u_{d-2}|_s &= 2\lambda_{d-2} + \gamma_{d-2} + \delta_{d-2} &&= 2\gamma_{d-3} + \lambda_{d-3} + \delta_{d-3} \\
|u_{d-1}|_s &= 2\lambda_{d-1} + \gamma_{d-1} + \delta_{d-1} &&= 2\gamma_{d-2} + \lambda_{d-2} + \delta_{d-2} \\
|u_d|_s &&&= 2\gamma_{d-1} + \lambda_{d-1} + \delta_{d-1}
\end{aligned}$$

As we know $z'_i = z_{i+1}$, which means $\gamma_i = \lambda_{i+1}\ \forall i \in \{1, 2, \ldots, d-2\}$. Now, from the above equations we have $\lambda_i + \delta_i = \gamma_{i+1} + \delta_{i+1}\ \forall i \in \{1, 2, \ldots, d-2\}$. Thus,

$$\begin{aligned}
|u_1|_s &= 2\lambda_1 + \gamma_1 + \delta_1 \\
&= \lambda_1 + \lambda_2 + \gamma_2 + \delta_2 && (\text{As } \lambda_1 + \delta_1 = \gamma_2 + \delta_2 \text{ and } \gamma_1 = \lambda_2) \\
&= \lambda_1 + \lambda_3 + \gamma_3 + \delta_3 && (\text{As } \lambda_2 + \delta_2 = \gamma_3 + \delta_3 \text{ and } \gamma_2 = \lambda_3) \\
&\quad \vdots \\
&= \lambda_1 + \lambda_{d-1} + \gamma_{d-1} + \delta_{d-1} && (\text{As } \lambda_{d-2} + \delta_{d-2} = \gamma_{d-1} + \delta_{d-1} \\
& && \qquad \text{and } \gamma_{d-2} = \lambda_{d-1}).
\end{aligned}$$

Now, $|u_1|_s - |u_d|_s = \lambda_1 + \lambda_{d-1} + \gamma_{d-1} + \delta_{d-1} - (2\gamma_{d-1} + \lambda_{d-1} + \delta_{d-1})$
$= \lambda_1 - \gamma_{d-1}$. Hence the proof.

□

Lemma 2.9. If u and v are two strings with $m_k(u) = m_k(v)$ and $\text{pref}_{k-1}(u) = \text{pref}_{k-1}(v)$ then $m_{k-1}(u) = m_{k-1}(v)$ and $\text{suff}_{k-1}(u) = \text{suff}_{k-1}(v)$.

Proof. From Corollary 2.6 we know that if the transformations which are used for getting v from u are of only these three forms k-tr_1, k-tr_2 or k-rt_2 then $m_{[1,k]}(u) = m_{[1,k]}(v)$. But if k-rt_1 is involved in those transformations then there is a possibility that $m_{k-1}(u) \neq m_{k-1}(v)$. But here we have $\text{pref}_{k-1}(u) = \text{pref}_{k-1}(v)$. From Lemma 2.8 we get that if k-rt_1 is applied on u any number of times to get v and s is any string with length less than k then

$$|\text{pref}_{k-1}(u)|_s - |\text{pref}_{k-1}(v)|_s = |u|_s - |v|_s.$$

Let $|s| = k-1$. As $\text{pref}_{k-1}(u)$ =$\text{pref}_{k-1}(v)$ then from the above equation we have $|u|_s = |v|_s$. Now in between k-rt_1 transformations any other types of k-transformations will not change the multiset factors upto length k. Hence, we see that any kind of transformations, if applied on u does not change the multiset of factors up to length k if $\text{pref}_{k-1}(u) = \text{pref}_{k-1}(v)$.

For the next part, the argument is that: all $k-$transformations act in a symmetric way on $(k-1)$ suffix and prefix while they are applied on the string. After applying a series of $k-$transformations if the prefix of size $k-1$ remains the same means that those transformations ensures the suffix of size $k-1$ remains the same.

□

Let $u \overset{k}{\equiv} v$. In order to obtain v from u we may need to apply one or more k-transformations. Assume $T = \langle t_1, t_2, \ldots, t_\eta \rangle$ be the sequence of k-transformations that are needed to get v from u if they are applied in the given order. Further, let $R = \langle r_1, r_2, \ldots, r_\psi \rangle$ be the subsequence of T such that r_i's are all k-rt_1 rotations. When a k-rt_1 rotation r_i is applied on a string u_{i-1} to get u_i, by Definition 2.3, u_{i-1} and u_i are of the form $z_{i-1}y_{i-1}z'_{i-1}y'_{i-1}z_{i-1}$ and $z_iy_iz'_iy'_iz_i$, respectively. We express the action of r_i as $u \xrightarrow[k\text{-}rt_1]{z_{i-1}z_i} v$.

Lemma 2.10. Let $u \overset{k}{\equiv} v$ and $u \overset{k-1}{\equiv} v$. Also, let T and R be as defined in the previous paragraph. If r_i is a transformation of the form $\xrightarrow[k\text{-}rt_1]{z_iz'_i}$ then

(a) $z'_i = z_{i+1}$ $\forall i \in \{1, 2, \ldots, (\psi - 1)\}$.

(b) $z_1 = z'_\psi$.

Proof. (a) This is an implication of the Corollary 2.6 (a). When a k-rt_1 rotation r_i is applied, the string z'_i becomes $(k-1)$-length suffix and prefix of u_i. From Lemma 2.6 (b), we know that the application of other k-transformations between any two k-rt_1 rotations, r_i and r_{i+1}, the string z'_i will remain the $(k-1)$-length suffix and prefix of intermediate strings. Thus, when the k-transformation r_{i+1} is used, z_i becomes z'_{i+1}. This happens for all the transformations indexed $i = 1$ to $\psi - 1$ in R.

(b) From the first proof we see that whenever a k-rt_1 rotation of the form $\xrightarrow[k\text{-}rt_1]{z_i z_{i+1}}$ is applied, the frequency of z_i is decreased by 1 and frequency of z_{i+1} is increased by 1. Now by Lemma 2.6, after the action of a k-rt_1 rotation any of the other three k-transformations will not change l-length substring multiset where $l \leq k$. Whenever the next k-rt_1 rotation is applied it must be of the form $\xrightarrow[k\text{-}rt_1]{z_{i+1} z_{i+2}}$. Thus frequency of z_{i+1} is restored. This is applicable for all transformations indexed from $i = 1$ to $\psi - 1$ in R. Thus, starting from u, the frequency of every of substring $z_2, z_3, \ldots, z_\psi$ is restored in v. But z_1 and $z_{\psi+1} = z'_\psi$ has not been restored yet. If $z_1 \neq z_{\psi+1}$ then the frequency of z_1 will remain one less in string u than in string v. Similarly, the frequency of z_ψ will become one more in string u than in string v. This cannot happen as it is assumed that $u \overset{k-1}{\equiv} v$. Hence, $z_1 = z'_\psi$ which means if $u \overset{k}{\equiv} v$ and $u \overset{k-1}{\equiv} v$ then $\text{pref}_{k-1}(u) = \text{pref}_{k-1}(v)$.

□

The proof of Lemma 2.10 (b) gives us an idea about the relation between z_1 and z'_ψ. The contrapositive of the second part of the above lemma is as follows.

Corollary 2.11. Let u and v be two distinct strings such that $\text{pref}_k(u) \neq \text{pref}_k(v)$ or $\text{suff}_k(u) \neq \text{suff}_k(v)$ and $k > 1$, then there exists a string x of length $\leq k + 1$ such that $|u|_x \neq |v|_x$.

This answers the question *(see [40])* "Can we get two strings $w = r\theta s$ and $v = s\theta' r$ such that $r \neq s$, $|r| = |s| = k - 1$ and $v \overset{i}{\equiv} w \ \forall 1 \leq i \leq k$, where $k \geq 3$?" in negation.

Theorem 2.12. Let u and v be two words such that $u \overset{i}{\equiv} v$ for $i = k, k - 1$. Then, $u \overset{i}{\equiv} v$ for all $1 \leq i \leq k$.

Proof. Here the assumptions are the same as in Lemma 2.10. Thus we have $\text{pref}_{k-1}(u) = \text{pref}_{k-1}(v) = z_1$. Let s be any string of length less than k. The other three k-transformations that are applied in between any two k-rt_1 rotations will not change the multiset of substrings of any length $\leq k$. We can apply Lemma 2.8 here and hence we get $|u|_s - |v|_s = |z_1|_s - |z_1|_s = 0$. This implies that there in no string of length less than $k - 1$ whose frequency can be different in u and v. Also, the

assumption was $u \overset{i}{\equiv} v$ for $i = k, k-1$. Combining this, we have $u \overset{i}{\equiv} v$ for all $1 \leq i \leq k$. □

This answers the question *(see [40])* "Does there exist two words w and v such that $u \overset{i}{\equiv} v$ for $i = k, k-1$ and $u \overset{k-2}{\not\equiv} v$?" in negation.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Chapter 3

Enumeration of k-Abelian Equivalent Words

3.1 Introduction

The relation between k-abelian equivalence and k-factor equivalence is discussed in the previous chapter. The properties of pair of words which are equivalent in either of the above two manner is studied. Karhumaki [29] introduced k-abelian equivalence to measure complexity of infinite words. By complexity we mean the number of equivalence classes created by the k-abelian equivalence relation on the set of a fixed length string on a particular finite alphabet. For any string, the cardinality of the corresponding k-abelian equivalence class gives the measure of its ambiguity with regard to string-reconstruction from the multiset of its factors up to length k. That cardinality is referred to as k-abelian ambiguity of the string. We shall investigate two special values of k and try to find highest possible k-abelian ambiguity.

In this chapter Σ will be used for an arbitrary finite alphabet of size ≥ 2, u will be used for an arbitrary string in Σ^n unless otherwise mentioned.

For the trivial case $k = 1$, the two multisets $m_k(u), m_{[1,k]}(u)$ are the same and $\max_{u\in\Sigma^n} |\mathbb{M}_1(u)| = \max_{0\leq r\leq n} {}^nC_r$ which is in $\Theta(\frac{2^n}{\sqrt{n}})$ [13]. For $k = 2$ we shall find the asymptotic value of $\max_{u\in\Sigma^n} |\mathbb{M}_{[1,2]}(u)|$ using Hutchinson and Wilf's theorem [24] on counting multiset permutations with adjacency information. We also show that this theorem

can be used to find the exact value of $|\mathbb{M}_{[1,k]}(u)|$ using the information obtained from $m_{[1,k]}(u)$.

It has been already shown in [40] that $|\mathbb{M}_{[1,k]}(u)| = 1$ for $k \geq \lfloor \frac{n}{2} \rfloor + 1$. The other special case we will be considering is $k = \lfloor \frac{n}{2} \rfloor$. In [40] we also have the characterization of the strings in an equivalence class for this value of k. We shall improve upon these results and obtain $\max_{u \in \Sigma^n} |\mathbb{M}_{[1,\lfloor \frac{n}{2} \rfloor]}(u)|$.

3.2 Background

3.2.1 De Bruijn Graph and Rauzy Multigraph

Definition 3.1 (De Bruijn graph [9] [16])**.** A directed graph G_k is said to be a De Bruijn graph of order k if it has 2^k vertices and each vertex is labelled as a k-length binary string. No two vertices are to be labelled with the same binary string. The graph has an edge from the vertex v_i to the vertex v_j if and only if the corresponding binary string at v_j has the same $(k-1)$-length prefix as the $(k-1)$ length suffix of the binary string at v_i.

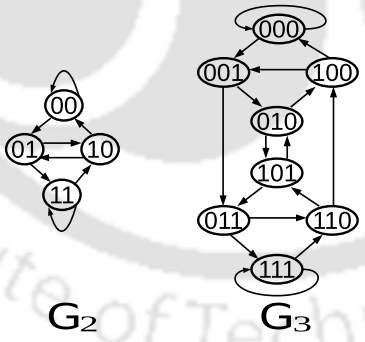

Figure 3.1: De Bruijn graph of order 2 and 3.

The edges of the graph G_k are labelled as $(k+1)$-length strings as shown in Figure 3.2. If e_{ij} is an edge from the vertex v_i to v_j with strings s_i and s_j respectively then e_{ij} must be labelled with the binary string $S(e_{ij}) = s_i \cdot \text{suff}_1(s_j)$ where '$\cdot$' is string concatenation operation.

A walk in a directed graph $G = (V, E)$ is an alternating sequence of edges and vertices starting and ending with a vertex such that left side and right side vertices of an edge are the starting and ending vertices of the edge. The walk is of length l if the number of edges in the sequence is l. Let $W_l = a_1w_1, a_2w_2, ..., a_lw_la_{l+1}$ be an walk where $\{w_1, w_2, ..., w_l\}$ and $\{a_1, a_2, ..., a_{l+1}\}$ are *multisets* and each $w_i \in E, a_i \in V$ such that the suffix of $S(w_i)$ is the same as prefix of $S(w_{i+1})$ $\forall i \in \{1, \ldots, l-1\}$. A walk E_l in a graph G_k gives us a string $s(W_l, G_k)$ of length $(l + k)$. We can express:

$$s(W_l, G_k) = S(w_1) \cdot \text{suff}_1(S(w_2)) \cdot \text{suff}_1(S(w_3)) \cdot \ldots \cdot \text{suff}_1(S(w_l)).$$

If the multiset $\{w_1, w_2, ..., w_l\}$ and the set E are equal, then the walk is known to be an Eulerian walk. There are two types of Eulerian walks:

1. Eulerian cycle or Eulerian circuit: Here $\text{pref}_k(S(w_1)) = \text{suff}_k(S(w_l))$.
2. Eulerian path: Here $\text{pref}_k(S(w_1)) \neq \text{suff}_k(S(w_l))$.

If we have an Eulerian walk $W_{2^{k+1}}$ on the graph G_k then $S(W_{2^{k+1}}, G_k)$ is called binary De Bruijn sequence [50] of order $(k+1)$. An m-ary De Bruijn sequence of order k has all possible k-length words (from an alphabet of size m) as factors with frequency exactly 1 for each of them. In Figure 3.2 we have De Bruijn graph G_3 with 4-length binary string labelled edges and an Eulerian walk E_{16} on G_3 gives a binary De Bruijn sequence 1100101101000011(110) of order 4. Here the De Bruijn sequence is of length 19 but as we can see its 3-length suffix and prefix are the same which is due to the nature of De Bruijn graph where an Eulerian walk becomes an Eulerian cycle. Thus a binary De Bruijn sequence of order k is usually expressed as a 2^{k+1}-length binary string removing the k-length suffix of the $(2^{k+1} + k)$-length string we get from an Eulerian walk on G_k.

From a De Bruijn graph G_k a De Bruijn sequence can easily be constructed but it may not be unique. Also we get back the De Bruijn graph G_k from any binary De Bruijn sequence of order $(k + 1)$ (without removing the k-length suffix).

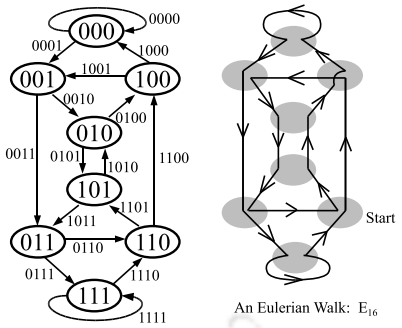

Figure 3.2: De Bruijn graph G_3 and an Eulerian walk E_{16}

Now consider an arbitrary string $u \in \Sigma^*$ with $|\Sigma| = \alpha$ and also the multiset of $(k+1)$-length factors of u, i.e., $m_{k+1}(u)$. Consider construction of a directed graph G with all k length factors of u as a vertex with an edge from vertex v_i to vertex v_j if and only if $S(e_{ij})$ is in $m_{k+1}(u)$. (Here the notation has the same meaning as discussed previously). As $m_{k+1}(u)$ is a multiset and there may be multiple occurrences of $S(e_{ij})$ in $m_{k+1}(u)$, we shall put as many edges from vertex v_i to vertex v_j as the number of occurrences of $S(e_{ij})$ is in $m_{k+1}(u)$. Hence the graph becomes a multigraph. We denote this graph as G_u^{k+1}. Let us call it **Rauzy multi graph** of the word u; a generalization of Rauzy graph [42].

If u is a binary De Bruijn sequence of order $(k+1)$(without removing the k-length suffix) then G_u^{k+1} is a De Bruijn graph of order k. Now if u is an arbitrary string in Σ^* then from G_u^{k+1} having an Eulerian walk will give us a string v which is of the same length as of u.

In addition $m_{k+1}(u) = m_{k+1}(v)$. So u is an arbitrary binary string and the only information we have about the string is $m_{k+1}(u)$. This is a way to reconstruct u and is the ideal version of DNA Sequencing By Hybridization (SBH) and this idea is discussed in [39].

3.2.2 Words with Given Adjacency Patterns

Finding number of Euler circuits in a graph is a well known problem. This problem was first discussed by Aardenne-Ehrenfest and de Bruijn [12] in 1951 and the formula

for enumerating number of Eulerian circuits in a graph is known as BEST theorem.

For a simple digraph different Eulerian paths always start at same vertex and also end at same vertex. An Eulerian circuit ends at the vertex where it started. These two properties were proved by I. J. Good [16] in 1946. The following two results are stated and proved [45]:

(i) A multidigraph D has an Eulerian cycle if and only if D is weakly connected up to isolated vertices and for every vertex, the in-degree equals the out-degree.

(ii) A multidigraph D has an eulerian path but no Eulerian cycle if and only if D is weakly connected up to isolated vertices and for all vertices with the possible exception of two, the in-degree equals the out-degree, and for those two vertices ((in-degree) − (out-degree)) will be exactly 1 and −1.

Finding the number of elements in the equivalence class generated by the relation $\overset{k}{\equiv}$ (introduced in 2.1), i.e. finding $|\mathbb{M}_k(u)|$, as we have seen, can be done using BEST theorem which involves finding rooted trees in a multidigraph. Counting Spanning Out-trees in Multidigraphs is done in an unpublished work [37]. Hutchinson and Wilf used BEST theorem to find the number of strings having the same given multiset of symbols with adjacency information. We will show in the next section that their result is useful in finding $|\mathbb{M}_{[1,k]}(u)|$ directly. We are stating their result in this regard in the following theorem:

Theorem 3.2. [24] Let $P = \{p_1, \ldots, p_t\}$ be a given set of positive integers, and let $Q = \left\{q_{(i,j)} | i, j \in \{1, \ldots, t\}\right\}$ be a given set of non-negative integers. If $\mathcal{N}(P, Q, t)$ is the number of strings that can be constructed from an ordered alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_t\}$ of t letters, in such a way that $\forall l \in \{1, 2, \ldots, t\}$ the letter $\sigma_l \in \Sigma$ appears exactly p_l times in the string and exactly $q_{(i,j)}$ times the letter σ_i is followed by the letter σ_j $\forall i, j \in \{1, 2, \ldots, t\}$ then:

$$\mathcal{N}(P, Q, t) = \left(\prod_{i=1}^{t}(p_i - 1)!\right)\left(\prod_{i,j=1}^{t} q_{(i,j)}!\right)^{-1} det\left[L(P, Q, t)\right]$$

Here $L(P, Q, t)$ is a matrix $(L_{ij})_{t\times t}$ defined by $L_{ij} = (p_i\delta_{ij} - q_{ij})$ and δ_{ij} is Kronecker delta function $\forall i, j \in \{1, \ldots, t\}$

3.3 Relation between $|\mathbb{M}_k(u)|$ and $|\mathbb{M}_{[1,k]}(u)|$

The information we obtain on a string u from $m_k(u)$ can also be obtained from $m_{[1,k]}(u)$. In addition to that $m_{[1,k]}(u)$ has much more information on u. Naturally the number of ambiguous reconstructions of u are same or fewer when $m_{[1,k]}(u)$ is provided in stead of $m_u(u)$ as a composition of u. The problem of reconstructing a string u using $m_{[1,k]}(u)$ is in fact more or less similar to that of using $m_k(u)$. An Eulerian walk in a Rauzy multigraph G_u^k gives us v which is an ambiguous reconstruction of u having the property $m_k(u) = m_k(v)$. Using a brute-force approach we consider all such graphs G_u^i and to find Eulerian walks in all of them and check if there is a common reconstruction which is appearing as Eulerian walk in every graph G_u^i. But the following result gives us simple condition which in addition with $m_k(u)$ gives us an equivalent information to $m_{[1,k]}(u)$. At the time of writing this chapter we found that this result has also been proved in [29]. We are giving an alternative proof.

Lemma 3.3. If u and v are two strings with $m_k(u) = m_k(v)$ and $\text{pref}_{k-1}(u) = \text{pref}_{k-1}(v)$ then $m_{[1,k]}(u) = m_{[1,k]}(v)$.

Proof. From Theorem 2.12 we get for two strings u and v if $m_k(u) = m_k(v)$ and $m_{k-1}(u) = m_{k-1}(v)$ then $m_{[1,k]}(u) = m_{[1,k]}(v)$.
It is also shown in Lemma 2.9 that under the given conditions we have $m_{k-1}(u) = m_{k-1}(v)$. Hence using these two results we have the proof. □

Given $m_k(u)$ and $m_{[1,k]}(u)$ for a string $u \in \Sigma^n$ the reconstruction, as we have seen, may not be unique. We use Eulerian walks on graphs to find these reconstructions. We see that there is a one-to-one relation between an Eulerian walk on multidigraph G_u^k and a reconstruction of u from $\mathbb{M}_k(u)$. So, enumeration of different Eulerian paths in G_u^k gives us the number of possible reconstruction of u from $m_k(u)$.

In both the cases where G_u^k has an Eulerian cycle or Eulerian path, the number of Eulerian circuit starting at the node which is induced by $(k-1)$-length prefix of u will give us the number of reconstructions of u from $m_{[1,k]}(u)$ as for the case of

Eulerian path the above two theorems ensure that all Eulerian paths in a multigraph will start at the same vertex and end at the same vertex.

Theorem 3.4. Given two strings $u, v \in \Sigma^n$:

(a) If $\text{suff}_{k-1}(u) \neq \text{pref}_{k-1}(u)$ and $m_k(u) = m_k(v)$ then $m_{[1,k]}(u) = m_{[1,k]}(v)$ and $|\mathbb{M}_k(u)| = |\mathbb{M}_{[1,k]}(v)|$

(b) If $\text{suff}_{k-1}(u) = \text{pref}_{k-1}(u)$ then $|\mathbb{M}_k(u)| = \eta|\mathbb{M}_{[1,k]}(v)|$ where η is the number of distinct $(k-1)$-length factors of u.

Proof. (a) If a string $u \in \Sigma^*$ is given with $m_{[1,k]}(u)$ then we construct the Rauzy multi graph G_k^u. Any Eulerian path in G_k^u will give us a reconstruction of u. The condition $\text{suff}_{k-1}(u) \neq \text{pref}_{k-1}(u)$ forces the Eulerian walk in G_k^u to be Eulerian path and not an Eulerian cycle. This also restricts every Eulerian path to start with the node induced by $\text{pref}_{k-1}(u)$ and to end with the node induced by $\text{suff}_{k-1}(u)$. The reconstruction v of u will not only have the property $m_k(u) = m_k(v)$ but also $m_{[1,k]}(u) = m_{[1,k]}(v)$ by Lemma 3.3. So, we have $|\mathbb{M}_k(u)| = |\mathbb{M}_{[1,k]}(v)|$.

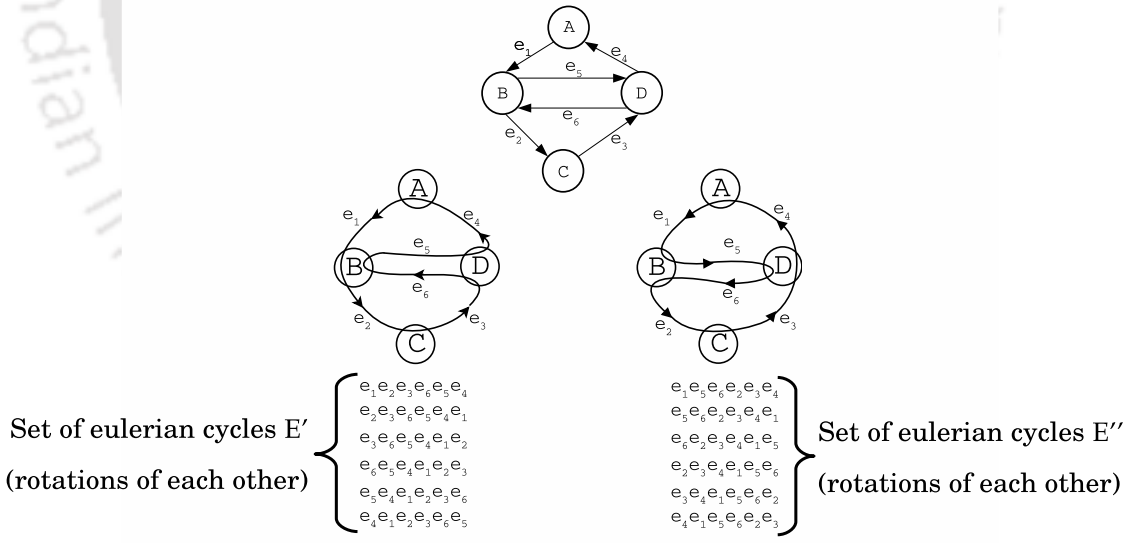

Figure 3.3: Two equivalence classes of Eulerian cycles with respect toa rotation.

(b) The condition $\text{suff}_{k-1}(u) = \text{pref}_{k-1}(u)$ implies that every Eulerian walk in G_k^u will be an Eulerian cycle. But two different Eulerian cycles may be rotations of each

other. The set of Eulerian cycles which are rotations of each other contains only one Eulerian cycle which will correspond to a reconstruction from $m_{[1,k]}(u)$. Every such set contains exactly η elements. But every element in those sets correspond to a different reconstruction from $m_k(u)$. Hence the result follows. □

If a string u is given with $m_{[1,k]}(u)$ then we know both $m_k(u)$ and $m_{k-1}(u)$ and vice versa by Theorem 2.12. If we consider the $(k-1)$-tuple notation of u which is $T_{(k-1)}(u)$ then any reconstruction v of u from $m_{[1,k]}(u)$ will have its $(k-1)$-tuple notation $T_{(k-1)}(v)$ and will be a permutation of $T_{(k-1)}(u)$ if we view $T_{(k-1)}(u)$ and $T_{(k-1)}(v)$ as strings in alphabet $\Sigma^u_{(k-1)}$ formed by different $(k-1)$-length factors of u. Here we can think $m_{k-1}(u)$ is expressing P in Theorem 3.2 if we replace Σ by $\Sigma^u_{(k-1)}$. Similarly $m_k(u)$ is expressing Q in Theorem 3.2 as adjacency information of $(k-1)$-length factors of u with multiplicity is embedded in $m_k(u)$. Hence using Theorem 3.2 we get the number of reconstructions of a string u from $m_{[1,k]}(u)$.

3.4 Asymptotic value of $\max_{u\in\Sigma^n} |\mathbb{M}_{[1,2]}(u)|$

Let us denote $\max_{u\in\Sigma^n} |\mathbb{M}_{[1,k]}(u)|$ as $A(\Sigma, n, k)$. We already know the values of $A(\Sigma, n, k)$ for $k = 1$ and $k \geq \lfloor \frac{n}{2} \rfloor + 1$. We investigate the value of $A(\Sigma, nk)$ for the intermediate values of k. In this section we consider the case $k = 2$. Using Theorem 3.2 we can calculate the $|\mathbb{M}_{[1,2]}|$ but to find $A(\Sigma, n, 2)$ we need to consider every u in Σ^n. We shall use the constraints given by Rauzy multigraph to take into consideration every possible u in $\Sigma = \{0, 1\}$ to give a lower bound on $A(\Sigma, n, 2)$ for any Σ.But before discussing that we shall prove the following combinatorial result which uses the maximum value of binomial coefficient and its conditions.

Theorem 3.5. $\max\left\{ \frac{(a+b)!}{a!b!} \frac{(b+d)!}{b!d!} \mid a + 2b + d = 4m \text{ and } a, b, d, m \in \mathbb{N} \cup \{0\} \right\}$ is in $\Theta\left(\frac{2^n}{n}\right)$ and maximum value occurs when $a = b = d = m$.

Proof. Let the maximum value occurs for some $n_1, n_2 \in \mathbb{N} \cup \{0\}$ such that $a + b = n_1$ and $b + d = n_2$. Now, $h_1 = \max\left\{ \frac{(a+b)!}{a!b!} \middle| a + b = n_1 \right\}$ is attained when $a = b$ if n_1 is even or $a, b \in \{\lfloor \frac{n_1}{2} \rfloor, \lfloor \frac{n_1}{2} \rfloor + 1\}$ with $a \neq b$ when n_1 is odd.

Similarly, $h_2 = \max\left\{ \frac{(b,d)!}{b!d!} \middle| b+d=n_2 \right\}$ is attained when $b=d$ if n_2 is even or $b,d \in \{\lfloor \frac{n_2}{2} \rfloor, \lfloor \frac{n_2}{2} \rfloor + 1\}$ with $b \neq d$ when n_2 is odd. It is to be noted that n_1 and n_2 has to be simultaneously even or odd as n_1+n_2 is even.

Let both n_1 and n_2 both are even. In that case to attain the maximum the relation $n_1 = n_2$ has to satisfy. Otherwise $\frac{(a+b)!}{a!b!}\ \frac{(b+d)!}{b!d!}$ will be $\leq h_1h_2$ which is attained when $a=b=d=m$. Hence, maximum value of $\frac{(a+b)!}{a!b!}\ \frac{(b+d)!}{b!d!}$ is $h_1h_2 = \frac{(2m)!(2m)!}{(m!)^4}$ and $a=b=d=m$.

Let, both n_1 and n_2 be odd. Without loss of generality let $n_1 < n_2$. Then to attain h_1h_2 we will have $a = \lfloor \frac{n_1}{2} \rfloor, b = \lfloor \frac{n_1}{2} \rfloor + 1 = \lfloor \frac{n_2}{2} \rfloor, d = \lfloor \frac{n_2}{2} \rfloor + 1$. Which means $a = m-1, b = m, d = m+1$. Then

$$\frac{h_1h_2}{\frac{(2m)!(2m)!}{(m!)^4}} = \frac{\frac{(2m-1)!(2m+1)!}{(m-1)!m!m!(m+1)!}}{\frac{(2m)!(2m)!}{(m!)^4}} = \frac{2m^2+m}{2m^2+2m} < 1$$

Hence the maximum is attained only when $a=b=d=m$. Now from the asymptotic value of central binomial coefficient we have $\frac{(2m)!}{m!m!}$ is $\Theta(\frac{4^m}{\sqrt{m}})$ [13].

We conclude that asymptotic value of $\frac{(2m)!(2m)!}{(m!)^4}$ is $\Theta(\frac{4^{2m}}{m})$ which is $\Theta(\frac{2^n}{n})$. □

Theorem 3.6. *$A(\Sigma, n, 2)$ is $\Omega(\frac{2^n}{n})$ for all finite alphabet Σ with $|\Sigma| \geq 2$.*

Proof. As $u \in \{0,1\}^n$ the Rauzy multigraph G_u^2 will have at most two nodes with label 0 and 1. Having only one node will lead us to the trivial case of unique reconstruction. Let G_u^2 is given by the Figure 3.4 where a, b, c, d are multiplicities of the edges $(0,0), (0,1), (1,0), (1,1)$ respectively and hence $a+b+c+d = n-1$.

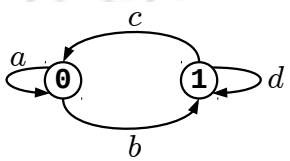

Figure 3.4: Rauzy multigrpah G_u^2

From Theorem 3.2 if p_1 corresponds to the symbol 0 and p_2 corresponds to the

symbol 1 then the matrix

$$L(P,Q,2) = \begin{bmatrix} p_1 - a & -b \\ -c & p_2 - d \end{bmatrix}$$

Now G_u^2 has to satisfy certain conditions to have an Eulerian path or Eulerian circuit. For that we must have $b = c$ or $|b - c| = 1$. There is no other restriction on a and d other than them being non negative integers.

Case $b = c$:

Without loss of generality, let the Eulerian walk starts from the node labelled by 0. Applying Theorem 3.2, we have $|u|_0 = p_1 = a+b+1$ (as $b = c$) and $|u|_1 = p_2 = b+d$. From Theorem 3.2, the number of reconstructions will be

$$\begin{aligned} \frac{(p_1-1)!(p_2-1)!}{a!b!b!d!}\ det\ \big(L(P,Q,2)\big) &= \frac{(a+b)!}{a!b!}\ \frac{(b+d-1)!}{b!d!}\ det \begin{bmatrix} b+1 & -b \\ -b & b \end{bmatrix} \\ &= \frac{(a+b)!}{a!b!}\ \frac{(b+d-1)!}{(b-1)!d!} \end{aligned}$$

We have from Theorem 3.5

$$\max \left\{ \frac{(a+b)!}{a!b!}\ \frac{(b+d-1)!}{(b-1)!d!} \middle| a,b,d \in \mathbb{N} \cup \{0\} \text{ and } a+2b+d = n-1 \right\} \text{ is in } \Theta\left(\frac{2^n}{n}\right)$$

Using symmetry we shall have the same result if we start the Eulerian walk from the node labelled as 1.

Case $|b - c| = 1$:

Let $b = c + 1$. In this scenario the eulerian walk has to start from the node labelled as 0. Then $p_1 = a + c + 1$ and $p_2 = d + c + 1$. The matrix

$$L(P,Q,2) = \begin{bmatrix} p_1 - a & -b \\ -c & p_2 - d \end{bmatrix} = \begin{bmatrix} c+1 & -c-1 \\ -c & c+1 \end{bmatrix}$$

From Theorem 3.2 the number of reconstructions will be:

$$\frac{(p_1-1)!(p_2-1)!}{a!(c+1)!c!d!}\ det\ \big(L(P,Q,2)\big) = \frac{(a+c)!}{a!c!(c+1)}\ \frac{(c+d)!}{c!d!}\ det \begin{bmatrix} c+1 & -c-1 \\ -c & c+1 \end{bmatrix}$$
$$= \frac{(a+c)!}{a!c!}\ \frac{(c+d)!}{c!d!}$$

Now similarly from Theorem 3.5 we have the maximum number of reconstructions in $\Theta\left(\frac{2^n}{n}\right)$.

Thus, we have for any binary string u, the highest number of reconstruction from $m_{[1,2]}(u)$ will be in $\Theta\left(\frac{2^n}{n}\right)$. Naturally if we extend this result it to any arbitrary alphabet Σ with $|\Sigma| \geq 2$ we have $A(\Sigma, n, 2)$ is $\Omega(\frac{2^n}{n})$. □

3.5 Finding $\max_{u\in\Sigma^n} |\mathbb{M}_{[1,k]}(u)|$ for $k = \lfloor \frac{|u|}{2} \rfloor$

For any string u, let us express the reverse of u by $\tilde{u}$. Intuitively as k increases, $\mathbb{M}_{[1,k]}(u)$ will decrease. Piña and Uzcátegui proved in [40] that if $k \geq \lfloor \frac{|u|}{2} \rfloor + 1$ then $\mathbb{M}_{[1,k]}(u) = 1$. They have also characterized the strings in $\mathbb{M}_{[1,k]}(u)$ for $k = \lfloor \frac{|u|}{2} \rfloor$ by the two following theorems:

Theorem 3.7. [40] If $u, v \in \Sigma^{2p}$ and $v \in \mathbb{M}_{[1,p]}(u)$ with $u \neq v$ for some alphabet Σ then u will be of the form $rab\tilde{r}$ for some $a, b \in \Sigma$, $r \in \{a, b\}^{p-1}$ and $v = \tilde{u}$.

Theorem 3.8. [40] If $u, v \in \Sigma^{2p+1}$ and $v \in \mathbb{M}_{[1,p]}(u)$ with $u \neq v$ for some alphabet Σ then u and v will be of the form:

(i) $u = wc$ and $v = \tilde{w}c$ where $w \in \{a, b\}^p$ with $a, b, c \in \Sigma$ and $m_{[1,p]}(w) = m_{[1,p]}(\tilde{w})$.

(ii) $u = cw$ and $v = c\tilde{w}$ where $w \in \{a, b\}^p$ with $a, b, c \in \Sigma$ and $m_{[1,p]}(w) = m_{[1,p]}(\tilde{w})$.

(iii) $u = rabc\tilde{r}$ and $v = \tilde{u}$ where $r \in \{a, b, c\}^p$ with $a, b, c \in \Sigma$.

(iv) $u = rabcs$ and $v = rcabs$ where $r, s \in \{a, b, c\}^p$ with $a, b, c \in \Sigma$.

(v) $u = rabcs$ and $v = rbcas$ where $r, s \in \{a, b, c\}^p$ with $a, b, c \in \Sigma$.

Lemma 3.9. [40] If $u, v \in \Sigma^{2p+1}$ for some alphabet Σ and $v \in \mathbb{M}_{[1,p]}(u)$ with $u = rabcs$ and $v = rbca$ where $r, s \in \Sigma^{p-1}$ and $a, b, c \in \Sigma$ then the following conditions hold:

(a) $r, s \in \{a, b, c\}^{p-1}$

(b) $s_1 \in \{a, b\}$

(c) $r_{k-i} = c \Leftrightarrow s_i = b \ \forall i \in \{1, \ldots, p-1\}$

(d) $r_{k-i} = a \Leftrightarrow s_i = a \ \forall i \in \{1, \ldots, p-1\}$

(e) $r_{k-i} = c \Leftrightarrow r_{k-i-1} = b \ \forall i \in \{1, \ldots, p-2\}$

(f) $r_{k-i} = b \Leftrightarrow r_{k-i-1} \in \{a, c\} \ \forall i \in \{1, \ldots, p-2\}$

(g) $r_{k-i} = a \Leftrightarrow r_{k-i-1} \in \{a, c\} \ \forall i \in \{1, \ldots, p-2\}$

From Theorem 3.7 it is clear that $\max_{u \in \Sigma^{2p}} |\mathbb{M}_{[1,p]}(u)| = 2$ because mirror image or reverse of a string is unique. In Theorem 3.8 the existence of ambiguous reconstruction of u from $\mathbb{M}_{[1,p]}(u)$ proves that $\max_{u \in \Sigma^{2p+1}} |\mathbb{M}_{[1,p]}(u)| \geq 2$.

We investigate if there can be a situation where $\max_{u \in \Sigma^{2p+1}} |\mathbb{M}_{[1,p]}(u)| > 2$.

In Theorem 3.8 all the five different forms of the pair u and v actually expresses five different permutations v of the symbols of the string u such that $m_{[1,k]}(u) = m_{[1,k]}(v)$ for $k = \lfloor \frac{|u|}{2} \rfloor$. If u and v is in the form (i) then let us say v is permutation $\pi_1(u)$, if u and v is in the form (ii) then let us say v is permutation $\pi_2(u)$ and so on.

If $v, x \in \mathbb{M}_{[1,k]}(u)$ and $v = \pi_j(u)$ and $x \neq u$ and $x \neq v$ then $x \neq \pi_j(u)$. So only 5 possible permutation of symbols in the string u are probable candidates to be a member of $\mathbb{M}_{[1,p]}(u)$.

Theorem 3.10. For $p \geq 2$, $\max_{u \in \Sigma^{2p+1}} |\mathbb{M}_{[1,p]}(u)| = 3$ and if $|\mathbb{M}_{[1,p]}(u)| = 3$ then u is a binary string.

We shall be needing the following few results to prove Theorem 3.10.

Lemma 3.11. Let $x \neq u, u \neq v, x \neq v$ with $v, x \in \mathbb{M}_{[1,p]}(u)$ and $v = \pi_1(u)$ then $x = \pi_2(u)$ with all three strings being binary.

Proof. From Theorem 3.7 and the form (i) in Theorem 3.8, let

$$u = q_1 R q_2 a b q_2 \tilde{R} q_1 c$$
$$v = q_1 R q_2 b a q_2 \tilde{R} q_1 c$$

where $q_1, q_2 \in \{a, b\}$ and $R \in \{a, b\}^{p-3}$ and $a, b, c \in \Sigma$.

If $p = 2$ then u will be of the form $qabqc$ and v is in the form $qbaqc$ where $q \in \{a, b\}$ and $a, b, c \in \Sigma$.

Case 1: $(x = \pi_2(u))$

Sub case 1.1: As x is in the form (ii) of Theorem 3.8, x must be $q_1 R q_2 a q_2 b \tilde{R} q_1 c$ by using Theorem 3.7. But from Theorem 3.7 this would not be possible if $c \notin \{a, b\}$. Here is an example with u, v and x in this form:

$$x = a\ aaaa\ ab\ aaaa$$
$$u = aaaa\ ab\ aaaa\ a$$
$$v = aaaa\ ba\ aaaa\ a$$

Sub case 1.2: If $p = 2$ then x has to be of the form $qbqac$. By Theorem 3.7, c also needs to be in $\{a, b\}$ to have x in this form.

Case 2: $(x = \pi_3(u))$

Sub case 2.1: As x is in the form (iii) of Theorem 3.8, x must be $c q_1 R q_2 b a q_2 \tilde{R} q_1$. So $q_1 R q_2 = c q_1 R$ which means $q_1 = q_2 = c$ and hence $u = x = c^p b c^p$, which is not possible.

Sub case 2.2: For $p = 2$, $x = qqbac$

Let $q = a$ then $u = x$, which is not possible.

Let $q = b$ then $x \notin \mathbb{M}_p(u)$, which is also not possible.

Case 3: $(x = \pi_4(u))$

Sub case 3.1: As x is in the form (iv) of Theorem 3.8, x must be $q_1Rq_2q_2ab\tilde{R}q_1c$.

Let $q_2 = a$. Then we have

$$u = q_1\ Raa\ ba\ \tilde{R}q_1c$$
$$u = q_1\ Raa\ ab\ \tilde{R}q_1c$$

Clearly $x \in \pi_2(u)$ and this situation has already been considered in the previous case.

Now, let $q_2 = b$. Then $x = v$, which is not possible.

Sub case 3.2: For $p = 2$, $x = qqabc$.

Let $q = a$ then $x \notin \mathbb{M}_p(u)$ which is also not possible.

Let $q = b$ then $v = x$, which is not possible.

Case 4: $(x = \pi_5(u))$

Sub case 4.1: As x is in the form (iv) of Theorem 3.8, x must be $q_1Rq_2bq_2a\tilde{R}q_1c$.

Let $q_2 = a$ then $x = v$, which is not possible.

Let $q_2 = b$ then $x = \pi_3(u)$ and this situation has already been considered in previous case.

Sub case 4.2: For $p = 2$, $x = qbqac$.

Let $q = a$ then $v = x$, which is not possible.

Let $q = b$ then $x \notin \mathbb{M}_p(u)$, which is also not possible.

Hence the only possible option for existence of u, v and x under given condition is $x = \pi_2(u)$ with all the three stings being binary. □

Lemma 3.12. Let u, v and x are three distinct strings with $v, x \in \mathbb{M}_{[1,p]}(u)$ and $v = \pi_2(u)$. Then $x = \pi_1(u)$ with all three strings being binary.

Proof. Cases where $x = \pi_1(u)$ are already taken care of in previous Lemma and $x \neq \pi_2(u)$ as $x \neq v$.

Case 1: $(x = \pi_3(u))$

We have $[m_{[1,p]}(u) = m_{[1,p]}(v)] \Leftrightarrow [m_{[1,p]}(\tilde{u}) = m_{[1,p]}(\tilde{v})]$. Thus

$$[v, x \in \mathbb{M}_{[1,p]}(u) \text{ and } v = \pi_2(u) \text{ and } x = \pi_3(u)]$$
$$\text{if and only if}$$
$$[\tilde{v}, \tilde{x} \in \mathbb{M}_{[1,p]}(\tilde{u}) \text{ and } \tilde{v} = \pi_1(\tilde{u}) \text{ and } \tilde{x} = \pi_3(\tilde{u})]$$

The later part is already taken care of and we have existence of different $\tilde{u}, \tilde{v}$ and $\tilde{x}$ means $\tilde{v} \in \pi_1(\tilde{u}), \tilde{x} \in \pi_2(\tilde{u})$ which implies $v \in \pi_2(u), x \in \pi_1(u)$. Also u, v and x are binary strings if and only if $\tilde{u}, \tilde{v}$ and $\tilde{x}$ are binary strings.

Case 2: $(x = \pi_4(u))$

We have $[m_{[1,p]}(u) = m_{[1,p]}(v)] \Leftrightarrow [m_{[1,p]}(\tilde{u}) = m_{[1,p]}(\tilde{v})]$. Thus

$$[v, x \in \mathbb{M}_{[1,p]}(u) \text{ and } v = \pi_2(u) \text{ and } x = \pi_4(u)]$$
$$\text{if and only if}$$
$$[\tilde{v}, \tilde{x} \in \mathbb{M}_{[1,p]}(\tilde{u}) \text{ and } \tilde{v} = \pi_1(\tilde{u}) \text{ and } \tilde{x} = \pi_5(\tilde{u})]$$

The later part has already been taken care of and we have existence of different $\tilde{u}, \tilde{v}$ and $\tilde{x}$ means $\tilde{v} \in \pi_1(\tilde{u}), \tilde{x} \in \pi_2(\tilde{u})$ which implies $v \in \pi_2(u), x \in \pi_1(u)$. Also u, v and x are binary strings if and only if $\tilde{u}, \tilde{v}$ and $\tilde{x}$ are binary strings.

Case 3: $(x = \pi_5(u))$

We have $[m_{[1,p]}(u) = m_{[1,p]}(v)] \Leftrightarrow [m_{[1,p]}(\tilde{u}) = m_{[1,p]}(\tilde{v})]$. Thus

$$[v, x \in \mathbb{M}_{[1,p]}(u) \text{ and } v = \pi_2(u) \text{ and } x = \pi_5(u)]$$
$$\text{if and only if}$$
$$[\tilde{v}, \tilde{x} \in \mathbb{M}_{[1,p]}(\tilde{u}) \text{ and } \tilde{v} = \pi_1(\tilde{u}) \text{ and } \tilde{x} = \pi_4(\tilde{u})]$$

The later part is already taken care of and we have existence of different $\tilde{u}, \tilde{v}$ and $\tilde{x}$ means $\tilde{v} \in \pi_1(\tilde{u}), \tilde{x} \in \pi_2(\tilde{u})$ which implies $v \in \pi_2(u), x \in \pi_1(u)$. Also u, v

and x are binary strings if and only if $\tilde{u}, \tilde{v}$ and $\tilde{x}$ are binary strings.

□

Lemma 3.13. Let u, v and x are three different strings with $v, x \in \mathbb{M}_{[1,p]}(u)$ and $v = \pi_3(u)$. Then v must be in the form of either $\pi_1(u)$ or $\pi_2(u)$ and $x = \pi_2(u)$ or $x = \pi_1(u)$ with all three strings being binary.

Proof. Cases where $x = \pi_1(u)$ or $x = \pi_2(u)$ were already taken care of and $x \neq \pi_3(u)$ as $x \neq v$.

Case 1: $(x = \pi_4(u))$

$$
\begin{aligned}
&[v, x \in \mathbb{M}_{[1,p]}(u) \text{ and } v = \pi_3(u) \text{ and } x = \pi_4(u)] \\
\Leftrightarrow\ &[u, x \in \mathbb{M}_{[1,p]}(v) \text{ and } u = \pi_3(v) \text{ and } x = \pi_2(v)] \\
\Leftrightarrow\ &[\tilde{u}, \tilde{x} \in \mathbb{M}_{[1,p]}(\tilde{v}) \text{ and } \tilde{x} = \pi_1(\tilde{v}) \text{ and } \tilde{u} = \pi_3(\tilde{v})] \\
\Rightarrow\ &[\tilde{u}, \tilde{x} \in \mathbb{M}_{[1,p]}(\tilde{v}) \text{ and } \tilde{x} = \pi_1(\tilde{v}) \text{ and } \tilde{u} = \pi_2(\tilde{v})], \text{ as } u \neq v, v \neq x, x \neq u \\
\Leftrightarrow\ &[u, x \in \mathbb{M}_{[1,p]}(v) \text{ and } u = \pi_1(v) \text{ and } x = \pi_2(v)] \\
\Leftrightarrow\ &[v, x \in \mathbb{M}_{[1,p]}(u) \text{ and } v = \pi_1(u) \text{ and } x = \pi_3(u)] \\
\Rightarrow\ &[v, x \in \mathbb{M}_{[1,p]}(u) \text{ and } v = \pi_1(u) \text{ and } x = \pi_2(u)], \text{ as } u \neq v, v \neq x, x \neq u
\end{aligned}
$$

Case 2: $(x = \pi_5(u))$

$$
\begin{aligned}
&[v, x \in \mathbb{M}_{[1,p]}(u) \text{ and } v = \pi_3(u) \text{ and } x = \pi_5(u)] \\
\Leftrightarrow\ &[u, x \in \mathbb{M}_{[1,p]}(v) \text{ and } u = \pi_3(v) \text{ and } x = \pi_1(v)] \\
\Rightarrow\ &[u, x \in \mathbb{M}_{[1,p]}(v) \text{ and } u = \pi_2(v) \text{ and } x = \pi_1(v)], \text{ as } u \neq v, v \neq x, x \neq u \\
\Leftrightarrow\ &[v, x \in \mathbb{M}_{[1,p]}(u) \text{ and } v = \pi_2(u) \text{ and } x = \pi_4(u)] \\
\Rightarrow\ &[v, x \in \mathbb{M}_{[1,p]}(u) \text{ and } v = \pi_2(u) \text{ and } x = \pi_1(u)]
\end{aligned}
$$

□

Lemma 3.14. Let u, v and x are three different strings with $v, x \in \mathbb{M}_{[1,p]}(u)$ and

$v = \pi_4(u)$. Then v must be in the form of either $\pi_1(u)$ or $\pi_2(u)$ and $x = \pi_2(u)$ or $x = \pi_1(u)$ with all three strings being binary.

Proof. Cases where $x = \pi_1(u)$ or $x = \pi_2(u)$ or $x = \pi_3(u)$ are already taken care of and $x \neq \pi_4(u)$ as $x \neq v$. So $x = \pi_5(u)$. From Theorem 3.8, we have $u = rabcs, v = rcabs$ and $x = rbcas$. Also we have $x \in \mathbb{M}_{[1,p]}(u)$ and $x = \pi_5(u)$. From Lemma 3.9, we get $r_{p-i} = c \Leftrightarrow s_i = b$ and $r_{p-i} = a \Leftrightarrow s_i = a \ \forall i = \{1, 2, \ldots, p-1\}$. Here $v \in \mathbb{M}_{[1,p]}(x)$ and $v = \pi_5(x)$. So using Lemma 3.9, we have $r_{p-i} = a \Leftrightarrow s_i = c$ and $r_{p-i} = b \Leftrightarrow s_i = b \ \forall i = \{1, \ldots, p-1\}$.

Let for some i, $s_i = a$. So, $r_{p-i} = a$ but that means $s_i = a$. Hence $c = a$ and u, v, x are binary strings. Also if $c = a$ then $x = \pi_1(u)$ and $v = \pi_2(v)$.

Let for some i, $s_i = b$. So, $r_{p-i} = c$. Also So, $r_{p-i} = b$. Hence, $b = c$ and u, v, x are binary strings. Also if $b = c$ then $x = \pi_3(u)$ and $v = \pi_1(u)$ which means $x = \pi_2(u)$ and $v = \pi_1(u)$ as $u \neq v, v \neq x$ and $x \neq u$.

Let for some i, $s_i = c$. So, $r_{p-i} = a$. Which means $s_i = a$. Hence $c = a$ and u, v, x are binary strings. Also if $c = a$ then $x = \pi_1(u)$ and $v = \pi_2(v)$. □

(Proof of Theorem 3.10) So, we have found that whenever we are choosing three different strings u, v and x of length $2p+1$ such that $m_{[1,p]}(u) = m_{[1,p]}(v) = m_{[1,p]}(x)$ then there is only two types of permutations of symbols of u are allowed to be v and x. Also all the three strings needed to be from an alphabet of size 2 to be in those forms. Hence we have the proof of Theorem 3.10.

3.6 Lower bound of $\max_{u\in\Sigma^n} |\mathbb{M}_{[1,k]}(u)|$ for $k = \Theta(\sqrt{n})$

We have an idea of the maximum number of reconstruction from $m_{[1,k]}$ for two special values of k. Here we have example of a class of strings which gives an idea about the maximum number of reconstructions when k is in $\Theta(\sqrt{n})$, which extends the investigation for the intermediate values of k in between 1 and $\lfloor \frac{|u|}{2} \rfloor + 1$. Here also we have used Theorem 3.2.

Lemma 3.15. $A(\Sigma, n, k)$ is $\Omega\left(\left(\frac{\sqrt{n}}{2}\right)^{\frac{\sqrt{n}}{2}}\right)$ when k is in $\Theta(\sqrt{n})$.

Proof. We are creating an example of a string in $\{0,1\}^n$.

Consider the string $u = (0^{2m}1)^m 0^{2m}$ of length $2m^2+3m$. From Table 3.6 we have multiset of factors. $(m-1)$-length factor multiset of u can be considered as

m- length factors		$(m-1)$- length factors	
factors	frequency	factors	frequency
0^m	$(m+1)^2$	0^{m-1}	$(m+1)(m+2)$
$0^{m-1}1$	m	$0^{m-2}1$	m
$0^{m-2}10$	m	$0^{m-3}10$	m
$0^{m-3}10^2$	m	$0^{m-4}10^2$	m
$0^{m-4}10^3$	m	$0^{m-5}10^3$	m
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$0^3 10^{m-4}$	m	$0^3 10^{m-5}$	m
$0^2 10^{m-3}$	m	$0^2 10^{m-4}$	m
010^{m-2}	m	010^{m-3}	m
10^{m-1}	m	10^{m-2}	m

Table 3.1: Multiset of m and $(m-1)$-length factors of $(0^{2m}1)^m 0^{2m}$

P in Theorem 3.2 and the adjacency information depicted by Q in Theorem 3.2, can be found in m-length factor multiset of u. Adjacency information which cannot be found in the multiset is taken as 0. For example $10^{m-2}1$ is not in u. So the corresponding value regarding adjacency of 10^{m-2} and $0^{m-2}1$ is zero.

The matrix $L(P,Q,t)$ will be:

$$\begin{bmatrix} m & -m & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & m & -m & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & m & -m & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & m & -m & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & -m & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & m & -m & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & m & -m \\ -m & -m & 0 & 0 & 0 & \dots & 0 & 0 & m \end{bmatrix}$$

Here $t = m$ and the determinant of the matrix is $2m^m$ and

$\left(\prod_{i=1}^{t}(p_i-1)!\right)\left(\prod_{i,j=1}^{t} q_{(i,j)}!\right)^{-1}$ can be written as:

$$\begin{aligned}\frac{((m^2+3m+1)!)\,((m-1)!)^{(m-1)}}{((m^2+2m+1)!)\,(m!)^m} &= \frac{\prod_{i=1}^{m}(m^2+2m+1+i)}{m!.m^{m-1}} \\ &= \frac{\prod_{i=1}^{m}(m+2+\frac{1+i}{m})}{(m-1)!} > \frac{m^m}{(m-1)!} \in \Omega(1) \\ &\quad \text{as } (m-1)! \in O(m^m)\end{aligned}$$

. Hence the total number of reconstructions of $(0^{2m}1)^m 0^{2m}$ will be in $\Omega(m^m)$. Now, from $n = 2m^2 + 3m$ we get:

$$m = \sqrt{\frac{8n+9}{16}} - \frac{3}{4} > \sqrt{\frac{8n}{16}} - 1 > \sqrt{\frac{n}{2}} - 1 > \frac{\sqrt{n}}{2}\ \forall n > 23. \tag{3.1}$$

Hence, $\left(\prod_{i=1}^{t}(p_i-1)!\right)\left(\prod_{i,j=1}^{t} q_{(i,j)}!\right)^{-1} \in \Omega\left(\left(\frac{\sqrt{n}}{2}\right)^{\frac{\sqrt{n}}{2}}\right)$

□

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Chapter 4

Multiset of k-length Factors with Full Precedence Information

4.1 Introduction

A generalization of k-abelian equivalence of words was recently done in [44] where the composition of words based on factors is given by multiset of all scattered sub words of length $\leq m$. The corresponding relation is called m-binomial equivalence.

We consider yet another extension of the idea of k-abelian equivalence where k-length factors are not only making permutation among themselves to give a different equivalent word but also keep the relative positions intact upto considering multiplicities of occurrences in the string as ordered pairs. In the context of DNA sequencing a similar problem has been studied. Using the idea of SBH, SNSH (Sequencing by Nested Strand Hybridization) has been developed [7, 43]. This procedure uses not only k-gram profile of a word but also their precedence order information. Rubinov and Gelfand [49] investigated a version of this problem where the aim is to reconstruct a single string. They gave a polynomial time algorithm to this problem. But their reconstruction did not take multiplicity of order information into consideration and also reconstruction length is arbitrary. Our approach to this problem is mainly combinatorial. The problem can also be visualized as a generalization of topological sorting where instead of a DAG (Directed Acyclic Graph) we are provided with a

directed multigraph which may contain cycles and loops.

4.2 Definitions and Notations

If $s \in \Sigma^n$ for some finite alphabet Σ, we denote the set of all distinct k-length factors of s as F_k^s, with factors ordered lexicographically so that we can refer to i^{th} k-length factor of s, $F_k^s(i)$, without ambiguity.

A factor can appear multiple number of times in a string. As mentioned in Chapter 2, the multiset of all factors of length k of a string s is denoted by $m_k(s)$. We can also think of organising the elements of $m_k(s)$ by sorting them lexicographically and without writing any element multiple times by attaching the frequencies with each element of F_k^s and denote the frequency array by m_k^s where $m_k^s(i) = |w|_s$ is used to indicate the frequency of a factor $w = F_k^s(i)$ in a string s.

As we have seen Chapter 2, $T_k(u)$ is the k-tuple notation of any given string u of length $\geq k$. We also write $T_k^{-1}(U) = u$ if there exists a string u such that $T_k(u) = U$.

Let w_1 and w_2 (not necessarily distinct) be factors of a string s that appears at location i and j in s, respectively. We say w_1 appears before w_2 in s if $i < j$.

For example, let $u = ababc$ then ab appears before ba (and also ab and bc) in u.

In this chapter we shall use the term *appears before* in the following sense:

Suppose u is a finite string from Σ^* where Σ is a finite alphabet. If u can be written as $u = u_1u_2 \dots u_n$ where $u_i \in \Sigma \ \forall i \in \{1, 2, \dots, n\}$. If v and w are two strings such that $v = u_pu_{p+1} \dots u_{p+k-1}$ and $w = u_qu_{q+1} \dots u_{q+l-1}$ and $p < q$, then we say that v *appears before* w in u as factors. Note that here $|v| = k \leq n$ and $|w| = l \leq n$. Also, given v, w and u we may get multiple different ordered pairs $(p_1, q_1), (p_2, q_2), \dots (p_r, q_r)$ similar to that of (p, q). In that case we say that v *appears before* w in u in r ways. Alternatively, we can say that v and w occur in u as ordered pair in r ways. An example is demonstrated in Figure 4.1.

Definition 4.1. Let s be a finite string over a finite alphabet Σ. Then, the k-factor precedence data or simply k-precedence data of s, $OP_k(s)$, is the multiset of ordered pairs (u, v) where u, v are factors of s and the frequency of (u, v) is defined as the number of different ways u appears before v in s.

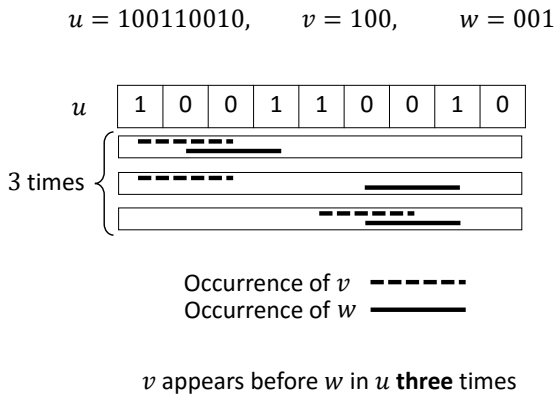

Figure 4.1: An example of a string v appears before w in u.

Two strings u and v are called k-precedence data equivalent if and only if $OP_k(u) = OP_k(v)$.

The frequency of (u, v) in $OP_k(s)$ is denoted by $OP_k^s(u, v)$ or $OP_k^s(i, j)$ or $|s|_{(u,v)}$, where $u = F_k^s(i)$ and $v = F_k^s(j)$.

Clearly we can express the frequencies by a matrix OP_k^s of size $r \times r$ where $r = |F_k^s|$. In other words the multiset $OP_k(s)$ can be expressed by the ordered pair (OP_k^s, F_k^s). For example, consider the string $s = 100111$. Then $F_2^s = \{00, 01, 10, 11\}$ and

$$OP_2^s = \begin{bmatrix} 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 2 \\ 1 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By precedence data we shall mean 1-precedence data.

4.3 Properties of Factor Precedence Multiset

In this section we explore various properties of OP_k^s.

Lemma 4.2. Let u and v be two strings from Σ^*. Then, $OP_k(u) = OP_k(v)$ implies $m_k(u) = m_k(v)$.

Proof. $OP_k^u = OP_k^v \Rightarrow \forall s \in \Sigma^k$, $OP_k^u(s, s) = OP_k^v(s, s)$. Observe that $OP_k^u(s, s)$ is

nothing but the number of ways to choose two distinct occurrences of s as a factor in u from $|u|_s$ such occurrences. So, $OP_k^u(s,s) = \frac{(|u|_s)^2 - |u|_s}{2}$.

Case $OP_k^u(s,s) > 0$**:** In this case, $|u|_s = |v|_s = \frac{1+\sqrt{1+8*OP_k^u(s,s)}}{2}$.

Case $OP_k^u(s,s) = 0$**:** If there exists a string s' such that either $OP_k^u(s,s') > 0$ or $OP_k^u(s',s) > 0$, then $|u|_s = |v|_s = 1$. Otherwise, $|u|_s = |v|_s = 0$.

This also shows that we can derive $m_k(u)$ from OP_k^u and hence we have the proof. □

Theorem 4.3. Let u and v be two strings from Σ^*. Then, $OP_k(u) = OP_k(v)$ implies $OP_i(u) = OP_i(v)\ \forall i \in \{1, \ldots, n\}$.

We prove the claim after proving the following results.

Lemma 4.4. Let Σ be a finite alphabet, $p, q \in \Sigma^{k-1}$ and $v \in \Sigma^*$. Then,

$$OP_{k-1}^v(p,q) = \sum_{a,b \in \Sigma} OP_k^v(pa, bq) + \mathcal{OL}_v(p,q)$$

$$\text{where, } \mathcal{OL}_v(p,q) = \begin{cases} |v|_{p.\text{suff}_1 q}, & \text{if } \text{suff}_{k-2}(p) = \text{pref}_{k-2}(q) \\ 0, & \text{otherwise} \end{cases}$$

Proof. The ordered pair (p,q) can be in the string v in the form pa and bq as k-length factors where pa is appearing before bq for all possible $a, b \in \Sigma$. The only other way (p,q) is in v, where that occurrence of (p,q) cannot be captured by precedence data of factors of length k, is when $\text{pref}_{k-2}(p) = \text{suff}_{k-2}(q)$. In that case $p.\text{suff}_1(q)$ contains (p,q). So counting all possible $(pa, bq) \in OP_k^v$ and $|v|_{p.\text{suff}_1 q}$ in case of $\text{pref}_{k-2}(p) = \text{suff}_{k-2}(q)$ gives us $OP_{k-1}^v(p,q)$. Here, $\mathcal{OL}_v(p,q)$ gives the frequency of $p.\text{suff}_1 q$ only when $\text{pref}_{k-2}(p) = \text{suff}_{k-2}(q)$. □

Example. If $v = 1001$, then we can easily verify that $OP_1^v(1,0) = 2$. But

$$\begin{aligned} & OP_2^v(\mathbf{1}0, 0\mathbf{0}) + OP_2^v(\mathbf{1}0, 1\mathbf{0}) + OP_2^v(\mathbf{1}1, 0\mathbf{0}) + OP_2^v(\mathbf{1}1, 1\mathbf{0}) \\ = \ & 1 + 0 + 0 + 0 \\ = \ & 1 \end{aligned}$$

Another occurrence of 1 appearing before 0 is hidden in the occurrence of the factor 10 as $\text{pref}_0(1) = \text{suff}_0(0) = \epsilon$. This is captured by $\mathcal{OL}_v(1,0) = 1$ and hence we get the correct value of $OP_1^v(1,0) = 1 + \mathcal{OL}_v(1,0) = 2$.

Lemma 4.5. Let Σ be a finite alphabet. Let $p, q \in \Sigma^{k-1}$ and $v \in \Sigma^*$. Then,

$$OP_{k-1}^v(p,q) = \sum_{a,b \in \Sigma} OP_k^v(ap, bq) + \mathcal{PR}_v(p,q)$$

$$\text{where, } \mathcal{PR}_v(p,q) = \begin{cases} |v|_q, & \text{if } p \text{ is prefix of } v \text{ and } p \neq q \\ |v|_q - 1, & \text{if } p \text{ is prefix of } v \text{ and } p = q \\ 0, & \text{otherwise} \end{cases}$$

Proof. The ordered pair (p,q) can be in the string v in the form ap and bq as k-length factors where ap is appearing before bq for all possible $a, b \in \Sigma$. The only other way (p,q) is in v, where that occurrence of (p,q) cannot be caught by occurrence of (ap, bq), is when p is a prefix of v. In that case the number of occurrences of (p,q) is missed the same number of times q is appears in v without being a prefix, which is $|v|_q$ if $p \neq q$ and $|v|_q - 1$ if $p = q$. So adding this number of missed cases with count of (ap, bq) in v, for all $a, b \in \Sigma$ gives us $OP_{k-1}^v(p,q)$. □

Example. If $v = 1000010001'$ then

$$\begin{aligned} & OP_2^v(00,01) = 8 = \\ & OP_2^v(\mathbf{0}00, \mathbf{0}01) + OP_2^v(\mathbf{0}00, \mathbf{1}01) + OP_2^v(\mathbf{1}00, \mathbf{0}01) + OP_2^v(\mathbf{1}00, \mathbf{1}01) \\ = \;& 5 + 0 + 3 + 0 = 8. \end{aligned}$$

But

$$\begin{aligned} & OP_2^v(10,01) = 3 \neq \\ & OP_2^(\mathbf{0}10, \mathbf{0}01) + OP_2^v(\mathbf{0}10, \mathbf{1}01) + OP_2^v(\mathbf{1}10, \mathbf{0}01) + OP_2^v(\mathbf{1}10, \mathbf{1}01) \\ = \;& 1 + 0 + 0 + 0 = 1. \end{aligned}$$

Here if we add $|v|_{01} = 2$, then we get the right answer.

Again,

$$\begin{aligned} & OP_2^v(10,10) = 1 \neq \\ & OP_2^(\mathbf{0}10,\mathbf{0}10) + OP_2^v(\mathbf{0}10,\mathbf{1}10) + OP_2^v(\mathbf{1}10,\mathbf{0}10) + OP_2^v(\mathbf{1}10,\mathbf{1}10) + |v|_{10} \\ = \; & 0+0+0+0+2 = 2. \end{aligned}$$

Here we have to subtract 1 from the obtained result as we are counting the prefix also while counting the occurrence after the prefix.

Lemma 4.6. Let Σ be an alphabet. Let $p, q \in \Sigma^{k-1}$ and $v \in \Sigma^*$. Then,

$$OP_{k-1}^v(p,q) = \sum_{a,b\in\Sigma} OP_k^v(pa,qb) + \mathcal{SF}_v(p,q)$$

$$\text{where, } \mathcal{SF}_v(p,q) = \begin{cases} |v|_p, & \text{if } q \text{ is suffix of } v \text{ and } p \neq q \\ |v|_p - 1, & \text{if } q \text{ is suffix of } v \text{ and } p = q \\ 0, & \text{otherwise} \end{cases}$$

Proof. The ordered pair (p, q) can be in the string v in the form pa and qb as k-length factors where pa is appearing before qb for all possible $a, b \in \Sigma$. The only other way (p, q) is in v, where that occurrence of (p, q) cannot be caught by occurrence of (pa, qb), is when q is a suffix of v. In that case the number of occurrences of (p, q) is missed the same number of times p is appears in v without being a suffix, which is $|v|_p$ if $p \neq q$ and $|v|_p - 1$ if $p = q$. So adding this number of missed cases with count of (pa, qb) in v, for all $a, b \in \Sigma$ gives us $OP_{k-1}^v(p, q)$. □

An example to demonstrate this result can be easily constructed by taking the reverse of the string which was taken to demonstrate the previous lemma.

Lemma 4.7. Let Σ be a finite alphabet. Let $p, q \in \Sigma^{k-1}$ and $v \in \Sigma^*$. Then,

$$OP_{k-1}^v(p,q) = \sum_{a,b\in\Sigma} OP_k^v(ap,qb) + \mathcal{PR}_v(p,q) + \mathcal{SF}_v(p,q) - \mathcal{MD}_v(p,q)$$

$$\text{where, } \mathcal{MD}_v(p,q) = \begin{cases} \beta - \delta - \gamma + |v|_p, & \text{if } p = q \\ \beta, & \text{otherwise} \end{cases}$$

$$\text{where, } \delta = \begin{cases} 1, & \text{if } p \text{ is prefix of } v \\ 0, & \text{otherwise} \end{cases}, \quad \text{and, } \gamma = \begin{cases} 1, & \text{if } q \text{ is suffix of } v \\ 0, & \text{otherwise} \end{cases}$$

$$\text{and, } \beta = \begin{cases} 1, & \text{if } p \text{ is prefix and } q \text{ is suffix of } v \\ 0, & \text{otherwise} \end{cases}$$

where $\mathcal{SF}_v(p,q)$ and $\mathcal{PR}_v(p,q)$ are as defined in Lemma 4.5 and 4.6, respectively.

Proof. The ordered pair (p,q) can be in the string v in the form ap and qb as k-length factors where ap is appearing before qb for all possible $a, b \in \Sigma$.

As we have seen in Lemma 4.5 and Lemma 4.6 we have to add both $\mathcal{PR}_v(p,q)$ and $\mathcal{SF}_v(p,q)$ to get the result as in this case count can be affected for p being prefix and q being suffix of v. But if $p = q$ then we are counting (p,q) twice while p is at prefix position and q be at suffix position. So, we need to subtract 1 thereafter. There are further terms to be subtracted. While calculating $\sum_{a,b\in\Sigma} OP_k^v(ap, qb)$ we have added the cases where for each occurrence of p $(= q)$ we have added some (ap, pb) to the list but here p and q coincides and cannot create a pair (p,q). Thus we have to subtract the number of such occurrences of p in v, that is, $|v|_p$, except the cases where p $(= q)$ is either suffix or prefix or both. These cases are taken care of by δ and γ and β. □

In Figure 4.2 and Figure 4.3 we present a demonstration of the Lemma 4.7 showing different cases that may occur while finding $OP_v^k - 1(p,q)$ from OP_v^k.

Proof. (*of Theorem 4.3*)

From Lemma 4.4 we get that for every $p, q \in \Sigma^{k-1}$, $OP_{k-1}^v(p,q)$ can be determined unambiguously from OP_k^v. Hence we get $OP_{k-1}(v)$ from $OP_k(v)$ and then we get $OP_{k-2}(v)$ from $OP_{k-1}(v)$ and so on. Hence we get $OP_i(v)$ from $OP_{i+1}(v)$ $\forall 1 \leq i \leq (k-1)$. Also we have $OP_k(u) = OP_k(v)$. So, in a similar manner we get $OP_{k-1}(u)$

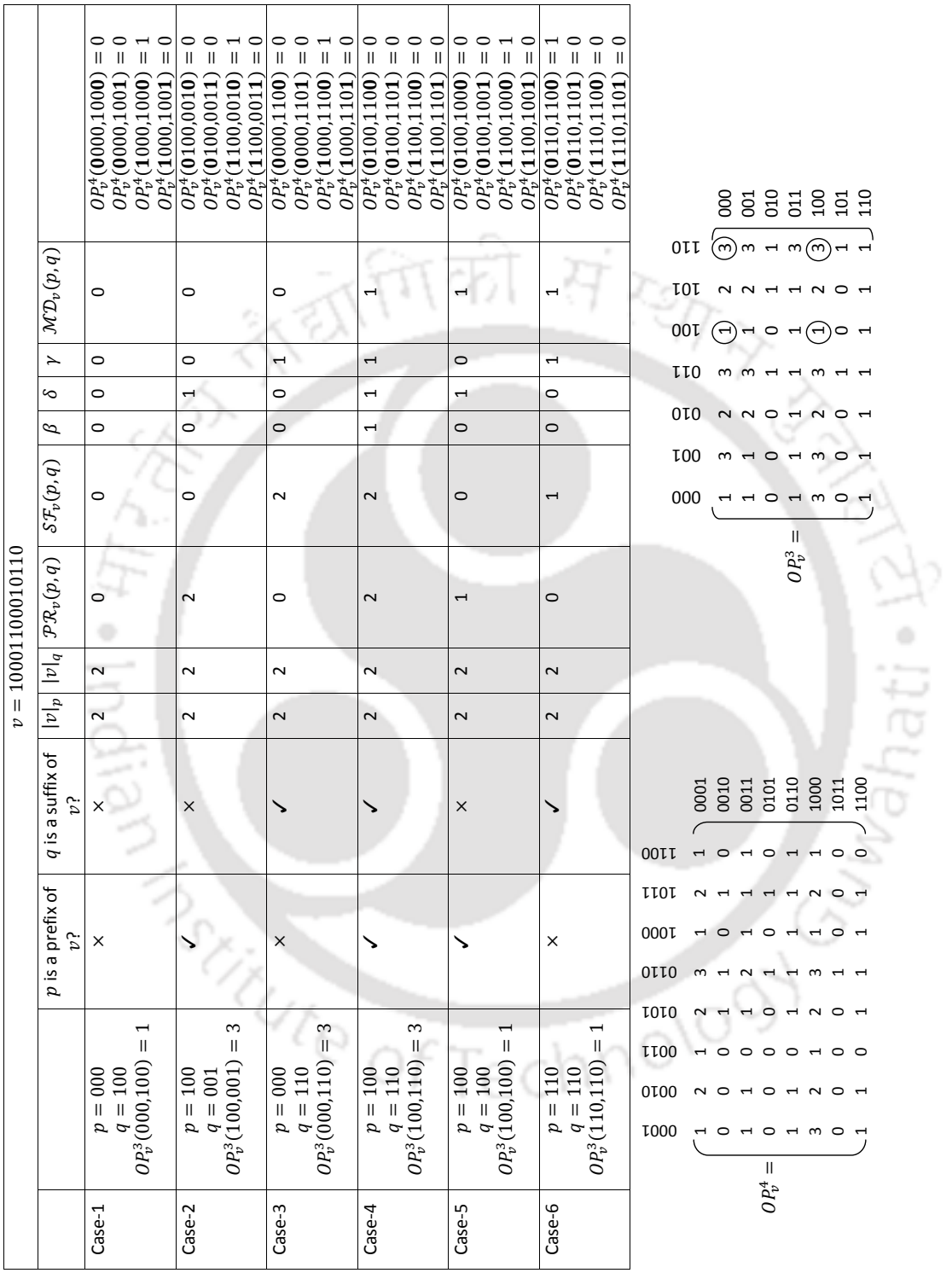

Figure 4.2: Examples of finding $OP_v^{k-1}(p,q)$ from OP_v^k.

$v = 1001101001001100100$

$p = 100$

$OP_v^3(100,100) = 1$

p is a prefix of v?	p is a suffix of v?	$\lvert v\rvert_p$	$\mathcal{PR}_v(p,p)$	$\mathcal{SF}_v(p,p)$	β	δ	γ	$\mathcal{MD}_v(p,p)$	
✓	✓	5	4	4	1	1	1	4	$OP_v^4(\mathbf{0}100,100\mathbf{0}) = 0$ $OP_v^4(\mathbf{0}100,100\mathbf{1}) = 5$ $OP_v^4(\mathbf{1}100,100\mathbf{0}) = 0$ $OP_v^4(\mathbf{1}100,100\mathbf{1}) = 1$

$OP_v^4 =$

0010	0011	0100	0110	1001	1010	1100	1101	
1	1	3	1	2	0	1	0	0010
3	1	4	3	4	1	2	1	0011
3	2	3	2	5	0	2	0	0100
3	1	4	1	4	1	2	1	0110
6	4	7	4	6	1	3	1	1001
2	1	3	1	3	0	1	0	1010
1	0	1	0	1	0	0	0	1100
2	1	3	1	3	1	1	0	1101

$OP_v^3 =$

001	010	011	100	101	110	
6	7	4	10	1	4	001
5	3	2	8	0	2	010
4	4	1	6	1	3	011
10	7	4	(10)	1	4	100
3	3	1	4	0	1	101
4	4	1	6	1	1	110

S

Figure 4.3: Examples of finding $OP_v^{k-1}(p,q)$ from OP_v^k where $p = q$ and p is both suffix and prefix of v.

from $OP_k(u)$ and then we get $OP_{k-2}(u)$ from $OP_{k-1}(u)$ and so on. Hence we have the result. □

Corollary 4.8. Let $u, v \in \Sigma^*$ for some finite alphabet Σ, then $OP_k^v = OP_k^v$ implies for all $m_{[1,k]}(u) = m_{[1,k]}(v)$.

Proof. By Theorem 4.3, $\forall i \in \{1, ..., k\}$ we have $OP_i^v = OP_i^v$ and by Lemma 4.2 we have now $m_{[1,k]}(u) = m_{[1,k]}(v)$. □

Corollary 4.9. Let u and v be two strings from Σ^*. Then, $OP_k(u) = OP_k(v)$ implies $\text{pref}_{k-1}(u) = \text{pref}_{k-1}(v)$ and $\text{suff}_{k-1}(u) = \text{suff}_{k-1}(v)$.

Proof. Lemma 4.2 and Corollary 4.8 together gives us the result. An alternative way to prove it is to look for a $p, q \in \Sigma^{k-1}$ such that $OP_{k-1}^u(p,q) \neq \sum_{a,b\in\Sigma} OP_k(ap, bq)$ Such p will be prefix of u. The same can be done with v and as $OP_k(u) = OP_k(v)$ we shall get the same prefix.

For suffix part, we need to look for again $p, q \in \Sigma^{k-1}$ such that $OP_{k-1}^u(p,q) \neq \sum_{a,b\in\Sigma} OP_k(pa, qb)$ Then q will be suffix for both u and v. □

Lemma 4.10. Let $u \in \Sigma^*$ and $v, w \in \Sigma^k$ with $v \neq w$. Then,

$$OP_k^u(v, w) + OP_k^u(w, v) = |u|_v |u|_w$$

Proof. The value $OP_k^u(v, w)$ is the number of ways v appears before w in u, and $OP_k^u(w, v)$ is the number of ways w appears before v in u. Together they form the statement number of ways v and w occurs anywhere in u as an unordered pair which is nothing but $|u|_v |u|_w$. □

The above results are all necessary conditions of a given precedence information to be valid but we have not arrived at any sufficient condition.

4.4 Reconstruction of u from $OP_k(u)$

The reconstruction of a word from the multiset of its factors is already discussed in Chapter 3. We have seen in the previous section that the same k-factor precedence data of two words implies their k-abelian equivalence. We shall use the existing algorithm to reconstruct a word from the multiset of factors and develop heuristics to reconstruct a word from the factor precedence data of k-length factors.

The first step is to form the Rauzy multigraph from the given precedence data which is straight forward from Lemma 4.2. We shall modify the Hierholzer algorithm [22] discussed in [28] for finding all Eulerian paths (or cycles) in the Rauzy multigraph. One way to find the reconstructions from k-precedence data is to scan through all these strings in $\mathbb{M}_k(u)$ and match the given k-precedence data. But that may take exponential time complexity as we have seen in Chapter 3 that number of reconstructions from k-length factor multiset can be exponential. Using the necessary conditions of a matrix to be the k-precedence data of a valid string we can modify the Eulerian path (cycle) finding algorithm. After forming the Rauzy multigraph from the given k-precedence data we can easily check in polynomial time if there is an Eulerian path or not. Now from the k-precedence data we can find the initial vertex of the Rauzy multigraph and then start removing one edge at a time and at each step modify the precedence matrix accordingly. We can check for the

validity of the precedence data using the necessary conditions. This process also may take exponential amount of time in the worst case but it may perform better when it is already known that $|\mathbb{M}_k(u)|$ is exponential.

An interesting question is if there is any polynomial time computational procedure to verify that a given matrix represents k-precedence data of some string.

To deal with this problem we have to look into the more basic problems of reconstructing a word from 1-precedence data.

For any $(n \times n)$-matrix M, if it corresponds to a string u such that $OP_1^u = M$ then from Lemma 4.10 it must obey:

$$M_{ij} + M_{ji} = t_i \times t_j \text{ where } 2M_{ii} = t_i^2 - t_i \tag{4.1}$$

But only having this property does not ensure the existence of such a string. For example the matrix:

$$\begin{bmatrix} 6 & 28 & 0 \\ 0 & 21 & 35 \\ 20 & 0 & 10 \end{bmatrix}$$

does not correspond to any string of length 16 and alphabet size 3.

But if we have any 2×2 matrix with the property mentioned above then it will correspond to a string. It can be easily verified that the matrix of the form:

$$\begin{bmatrix} C & A \\ B & D \end{bmatrix}$$

where $2C = a^2 - a$ and $2D = b^2 - b$, will be a 1-precedence matrix of the string $1^{b-p-1}0^q10^{a-q}1^p$ where $p = \lfloor A/a \rfloor$ and $q = A \bmod a$. A given $r \times r$ matrix M, together with a lexicographically ordered set S of k-length strings with $|S| = r$, may not correspond to any string s such that $OP_k^s = M, F_k^s = S$ and $|F_k^s| = r$. We call such a matrix invalid with respect to k and S.

An invalid matrix with respect to given k and S may also satisfy Equation 4.1. One interesting question is how to compute in polynomial time if a matrix is invalid

with respect to given $k > 1$ and S. Answering this will immediately lead us to the method reconstruction of a string s from $OP_k(s)$ in polynomial time.

If we assume that a given matrix M is valid and there is a string s corresponding to M, then the matrix OP^s_{k-1} formed using M following the result given by Lemma 4.4. In case of an invalid matrix M with respect to a given k and S, the matrix M' if constructed using the same way OP^s_{k-1} was made then M' will be invalid with respect to $k-1$ and S' but may satisfy Equation 4.1 where S' is the lexicographically ordered set of $(k-1)$-length strings constructed from S using Lemma 4.4.

To summarize this section, if precedence data is used as as extra check in the process of reconstructing a string from its multiset of factors of lengths up to k, then we shall get a better performance. But if we proceed to reconstruct a string from its k- precedence data only, then exponential running time may arise as it is unknown that if checking a validity of a matrix from the context of k-precedence data can the done in polynomial time.

4.5 Permutation in Multiset with the Same Precedence Data

For any two strings u and v, whenever we have $OP_k(u) = OP_k(v)$, there is a permutation involved in the k-tuple representation of u to obtain another k-tuple representation which gives us v. But these permutations are of special kind apart from being on a multiset. They also keep the precedence data identical in the elements of the multiset. To get an insight about these permutations we consider the simple case of $k = 1$. The following results gives sufficient conditions for a pair of strings to have the same precedence data.

Lemma 4.11. If $u, v \in \Sigma^n$ for some finite alphabet Σ and $u = UU'$ and $v = VV'$ for some $U, U', V, V' \in \Sigma^*$ such that $OP_1(U) = OP_1(V)$ and $OP_1(U') = OP_1(V')$ then $OP_1(u) = OP_1(v)$.

Proof. Consider any pair of symbols or 1-length strings $s, r \in \Sigma$. They occur in u as an ordered pair in three different ways. These are

(i) s, r are both in U,

(ii) s, r are both in U',

(iii) s in U and r in U'.

So, $OP_1^u(s,r) = OP_1^U(s,r) + OP_1^{U'}(s,r) + |U|_s \times |U'|_r$.
Similarly, $OP_1^v(s,r) = OP_1^V(s,r) + OP_1^{V'}(s,r) + |V|_s \times |V'|_r$. According to the given conditions $OP_1^U(s,r) = OP_1^V(s,r)$ and $OP_1^{U'}(s,r) = OP_1^{V'}(s,r)$ and also from Corollary 4.8 we have $|U|_s = |V|_s$ and $|U'|_r = |V'|_r$. Hence $OP_1(u) = OP_1(v)$. □

Lemma 4.12. If $u, v \in \Sigma^n$ for some finite alphabet Σ with u being in the form:

$$u = U_0W_1U_1W_2U_2W_3U_3W_4U_4$$

and v is in any of the following forms:

(a) $V_0W_2'V_1W_1'V_2W_4'V_3W_3'V_4$,

(b) $V_0W_3'V_1W_1'V_2W_4'V_3W_2'V_4$,

(c) $V_0W_2'V_1W_4'V_2W_1'V_3W_3'V_4$,

(d) $V_0W_3'V_1W_4'V_2W_1'V_3W_2'V_4$.

for some $U_i, V_i, W_i, W_i' \in \Sigma^*$ and satisfying the following conditions:

(i) $m_1(U_1) = m_1(U_3)$

(ii) $m_1(W_1) = m_1(W_4)$

(iii) $m_1(W_2) = m_1(W_3)$

(iv) $OP_1(U_i) = OP_1(V_i)$ $\forall i \in \{0,1,2,3,4\}$

(v) $OP_1(W_i) = OP_1(W_i')$ $\forall i \in \{1,2,3,4\}$

then $OP_1(u) = OP_1(v)$.

Proof. From the given conditions we have $OP_1(V_1) = OP_1(U_1) \Rightarrow m_1(V_1) = m_1(U_1) = m_1(U_3)$. Also, $OP_1(V_3) = OP_1(U_3) \Rightarrow m_1(V_3) = m_1(U_3)$. So, we have

$$m_1(V_3) = m_1(U_3) = m_1(V_1) = m_1(U_1) \tag{4.2}$$

Similarly from $(ii), (iii)$ and (v) we have

$$m_1(W_1) = m_1(W_4) = m_1(W_1') = m_1(W_4') \tag{4.3}$$
$$m_1(W_2) = m_1(W_3) = m_1(W_2') = m_1(W_3') \tag{4.4}$$

Now let,

$$u' = W_1U_1W_2U_2W_3U_3W_4$$
$$v' = W'_{q_1}V_1W'_{p_1}V_2W'_{p_2}V_3W'_{q_2}$$

where $p_1, p_2 \in \{1, 4\}$ and $q_1, q_2 \in \{2, 3\}$ with $p_1 \neq p_2, q_1 \neq q_2$.

Let $r, s \in \Sigma$. Then

$$\begin{aligned} OP_1^{u'}(r,s) = & \sum_{i=1,2,3} OP_1^{U_i}(r,s) + \sum_{i=1,2,3,4} OP_1^{W_i}(r,s) \\ & + |W_1|_r \left(\sum_{i=1,2,3} |U_i|_s + \sum_{i=2,3,4} |W_i|_s \right) \\ & + |W_2|_r \left(\sum_{i=2,3} |U_i|_s + \sum_{i=3,4} |W_i|_s \right) + |W_3|_r \big(|U_3|_s + |W_4|_s \big) \\ & + |U_1|_r \left(\sum_{i=2,3} |U_i|_s + \sum_{i=2,3,4} |W_i|_s \right) + |U_3|_r |W_4|_s \\ & + |U_2|_r \left(|U_3|_s + \sum_{i=3,4} |W_i|_s \right) \end{aligned}$$

$$\begin{aligned} \Rightarrow OP_1^{u'}(r,s) = & \sum_{i=1,2,3} OP_1^{V_i}(r,s) + \sum_{i=1,2,3,4} OP_1^{W_i}(r,s) \\ & + |W_1|_r \left(2|v_1|_s + |v_2|_s + |W_1|_s + 2|W_2|_s\right) \\ & + |W_2|_r \left(2|v_1|_s + |v_2|_s + 2|W_1|_s + |W_2|_s\right) \\ & + |V_1|_r \left(|v_1|_s + |v_2|_s + 2|W_1|_s + 2|W_2|_s\right) \\ & + |V_2|_r \left(|v_1|_s + |W_1|_s + |W_2|_s\right) \end{aligned}$$

Now,

$$
\begin{aligned}
OP_1^{v'}(r,s) = &\sum_{i=1,2,3} OP_1^{V_i}(r,s) + \sum_{i=1,2} OP_1^{W'_{p_i}}(r,s) + \sum_{i=3,4} OP_1^{W'_{q_i}}(r,s) \\
&+ |W'_{q_1}|_r \left(\sum_{i=1,2,3} |V_i|_s + \sum_{i=1,2} |W'_{p_i}|_s + |W'_{q_2}|_s \right) \\
&+ |W_{p_1}|_r \left(\sum_{i=2,3} |V_i|_s + |W'_{p_2}|_s + |W'_{q_2}|_s \right) + |W'_{p_2}|_r \big(|V_3|_s + |W'_{q_2}|_s \big) \\
&+ |V_1|_r \left(\sum_{i=2,3} |V_i|_s + \sum_{i=1,2} |W_{p_i}|_s + W'_{q_2}|_s \right) + |V_3|_r |W_{q_2}|_s \\
&+ |V_2|_r \left(|V_3|_s + |W_{p_2}|_s + + |W_{q_2}|_s \right)
\end{aligned}
$$

$$
\begin{aligned}
\Rightarrow OP_1^{u'}(r,s) = &\sum_{i=1,2,3} OP_1^{V_i}(r,s) + \sum_{i=1,2,3,4} OP_1^{W_i}(r,s) \\
&+ |W_1|_r \left(2|v_1|_s + |v_2|_s + |W_1|_s + 2|W_2|_s \right) \\
&+ |W_2|_r \left(2|v_1|_s + |v_2|_s + 2|W_1|_s + |W_2|_s \right) \\
&+ |V_1|_r \left(|v_1|_s + |v_2|_s + 2|W_1|_s + 2|W_2|_s \right) \\
&+ |V_2|_r \left(|v_1|_s + |W_1|_s + |W_2|_s \right) \\
&\text{[Using the given conditions and Equations (4.2), (4.3) and (4.4)]}
\end{aligned}
$$

So, we have $OP_1(u') = OP_1(v')$. Again $u = U_0 u' U_4$ and $v = V_0 v' V_4$ with $OP_1(U_0) = OP_1(V_0)$ and $OP_1(U_4) = OP_1(V_4)$. Applying Lemma 4.11 we have $OP_1(u) = OP_1(v)$. □

The smallest pair of strings which can be obtained by applying transposition to another is of the form *ab* and *ba*. It can be easily verified that the smallest pair of distinct strings having the same precedence data is of the form *abba* and *baab*. One may see that the sufficient condition in Lemma 4.12 is inspired by this form. Again conditions (iv) and (v) in Lemma 4.12 are recursive in nature. They allow chunks of the string to be different but with the same precedence data. We consider the case where $U_i = V_i, W_i = W'_i$ for all i and define the following:

Definition 4.13 (Mirror transposition)**.** If $u, v \in \Sigma$ for any finite alphabet Σ and u is of the form:

$$u = U_0W_1U_1W_2U_2W_3U_3W_4U_4$$

then v is said to be obtained by using mirror transposition if and only if v is in the form:

$$v = U_0W_{q_1}U_1W_{p_1}U_2W_{p_2}U_3W_{q_2}U_4$$

for some $p_1, p_2 \in \{1, 4\}$, $q_1, q_2 \in \{2, 3\}$ with $p_1 \neq p_2$ and $q_1 \neq q_2$ and $U_i, W_i \in \Sigma^*$ satisfying

(i) $m_1(U_1) = m_1(U_3)$ (ii) $m_1(W_1) = m_1(W_4)$ (iii) $m_1(W_2) = m_1(W_3)$

One can observe that if v can be obtained by applying mirror transposition on u and vice versa. We say $u \underset{m}{\equiv} v$ if and only if v can be obtained by using mirror transposition on u. The relation $\underset{m}{\equiv}$ is reflexive, symmetric but not transitive. We denote the transitive closure of the relation by $\overset{+}{\underset{m}{\equiv}}$. Whether the two sets $\{v | v \overset{+}{\underset{m}{\equiv}} u\}$ and $\{w | OP_1(w) = OP_1(u)\}$ for any arbitrary finite string u is true or not is still not known.

4.6 Smallest Pair of Strings with the Same k-Precedence Data

In Chapter 3 we have seen smallest pair of k-abelian equivalent strings have length $2k + 1$. In this section our goal is find an analogous result for pair of strings having same k-precedence data. In order to do so we need to understand the specific types of permutation a string undergo such that the resultant string will have the same k-precedence data. Pevzner [38] has shown that in order to achieve permutation on strings keeping k-length factor multiset the same, we need to consider the $(k-1)$-tuple representation of a string. Motivated by Definition 2.2 we define the following transformation:

Definition 4.14 (k-mirror transposition)**.** Let y and y' be two strings that can be expressed as either of these four forms given below.

$$(1^{st})\begin{bmatrix} T_{(k-1)}(y) &= y_1z_1y_2z_2y_3z_1y_4z_2y_5z_1y_6z_2y_7z_1y_8z_2y_9 \\ T_{(k-1)}(y') &= y_1z_1y_{q_1}z_2y_3z_1y_{p_1}z_2y_5z_1y_{p_2}z_2y_7z_1y_{q_2}z_2y_9 \end{bmatrix}$$

$$(2^{nd})\begin{bmatrix} T_{(k-1)}(y) &= y_1zy_2zy_3zy_4zy_6zy_7zy_8zy_9 \\ T_{(k-1)}(y') &= y_1zy_{q_1}zy_3zy_{p_1}zy_{p_2}zy_7zy_{q_2}zy_9 \end{bmatrix}$$

$$(3^{rd})\begin{bmatrix} T_{(k-1)}(y) &= y_1zy_2zy_4zy_5zy_6zy_8zy_9 \\ T_{(k-1)}(y') &= y_1zy_{q_1}zy_{p_1}zy_5zy_{p_2}zy_{q_2}zy_9 \end{bmatrix}$$

$$(4^{th})\begin{bmatrix} T_{(k-1)}(y) &= y_1zy_2zy_4zy_6zy_8zy_9 \\ T_{(k-1)}(y') &= y_1zy_{q_1}zy_{p_1}zy_{p_2}zy_{q_2}zy_9 \end{bmatrix}$$

where $|z_1| = |z_2| = |z| = k-1$, $y_i \in \Sigma^*$, $|y_i| = j_i(k-1)$ for some non negative integers j_i $\forall i \in \{1,\ldots,9\}$, $p_1, p_2 \in \{2,8\}$, $q_1, q_2 \in \{4,6\}$ with $p_1 \neq p_2, q_1 \neq q_2$ and the following three conditions are also satisfied:

1. $m_k(T_{k-1}^{-1}(z_2y_3z_1)) = m_k(T_{k-1}^{-1}(z_2y_7z_1))$ and $m_k(T_{k-1}^{-1}(zy_3z)) = m_k(T_{k-1}^{-1}(zy_7z))$
2. $m_k(T_{k-1}^{-1}(z_1y_2z_2)) = m_k(T_{k-1}^{-1}(z_1y_8z_2))$ and $m_k(T_{k-1}^{-1}(zy_2z)) = m_k(T_{k-1}^{-1}(zy_8z))$
3. $m_k(T_{k-1}^{-1}(z_1y_4z_2)) = m_k(T_{k-1}^{-1}(z_1y_6z_2))$ and $m_k(T_{k-1}^{-1}(zy_4z)) = m_k(T_{k-1}^{-1}(zy_6z))$

Then the two strings y and y' are k-mirror transpositions of each other.

We say $u \underset{mTr}{\overset{k}{\equiv}} v$ if and only if u is obtained by applying k-mirror transposition on v or vice-versa. Here also the relation $u \underset{mTr}{\overset{k}{\equiv}} v$ is reflexive and symmetric but not transitive. One can see that we have only taken into account transpositions but not something analogous to rotation in Definition 2.3. As $OP_k(u) = OP_k(v) \Rightarrow m_{[1,k]}(u) = m_{[1,k]}(v)$ from Chapter 2, the effect of two rotations keeping suffix and prefix of length $(k-1)$ the same, can be done using transpositions.

Now we have the definition of the k-mirror transposition we must establish that such transpositions will keep k-precedence data of a string.

Lemma 4.15. If $u \underset{mTr}{\overset{k}{\equiv}} v$ then $OP_k(u) = OP_k(v)$.

Proof. Consider the first form in Definition 4.14. Let us denote

- $Y_1 = T_k(T^{-1}_{(k-1)}(y_1z_1))$, $X_1 = T_k(T^{-1}_{(k-1)}(z_1y_2z_2))$
- $Y_2 = T_k(T^{-1}_{(k-1)}(z_2y_3z_1))$, $X_2 = T_k(T^{-1}_{(k-1)}(z_1y_4z_2))$
- $Y_3 = T_k(T^{-1}_{(k-1)}(z_2y_5z_1))$, $X_3 = T_k(T^{-1}_{(k-1)}(z_1y_6z_2))$
- $Y_4 = T_k(T^{-1}_{(k-1)}(z_2y_7z_1))$, $X_4 = T_k(T^{-1}_{(k-1)}(z_1y_8z_2))$
- $Y_5 = T_k(T^{-1}_{(k-1)}(z_2y_9))$

Then

$$T_k(y) = Y_1X_1Y_2X_2Y_3X_3Y_4X_4Y_5$$
$$T_k(y) = Y_1X_{g_1}Y_2X_{h_1}Y_3X_{h_2}Y_4X_{g_2}Y_5$$

where

$$f_1 = \begin{cases} 1 & \text{if } p_1 = 2 \\ 4 & \text{if } p_2 = 8 \end{cases}, \qquad g_1 = \begin{cases} 2 & \text{if } q_1 = 4 \\ 3 & \text{if } q_2 = 6 \end{cases}$$

with $f_1 \neq f_2, g_1 \neq g_2, f_1, f_2 \in \{1, 4\}$ and $g_1, g_2 \in \{2, 3\}$.

Consider the second, third and fourth form in Definition 4.14. Let us denote

- $Y_1 = T_k(T^{-1}_{(k-1)}(y_1z))$, $X_1 = T_k(T^{-1}_{(k-1)}(zy_2z))$
- $Y_2 = T_k(T^{-1}_{(k-1)}(zy_3z))$, $X_2 = T_k(T^{-1}_{(k-1)}(zy_4z))$
- $Y_3 = T_k(T^{-1}_{(k-1)}(zy_5z))$, $X_3 = T_k(T^{-1}_{(k-1)}(zy_6z))$
- $Y_4 = T_k(T^{-1}_{(k-1)}(zy_7z))$, $X_4 = T_k(T^{-1}_{(k-1)}(zy_8z))$
- $Y_5 = T_k(T^{-1}_{(k-1)}(zy_9))$

Then we have

$$2^{nd} \text{ form in Definition 4.14} \begin{cases} T_k(y) &= Y_1X_1Y_2X_2X_3Y_4X_4Y_5 \\ T_k(y) &= Y_1X_{g_1}Y_2X_{f_1}X_{f_2}Y_4X_{g_2}Y_5 \end{cases}$$

$$3^{rd} \text{ form in Definition 4.14} \begin{cases} T_k(y) &= Y_1X_1X_2Y_3X_3X_4Y_5 \\ T_k(y) &= Y_1X_{g_1}X_{f_1}Y_3X_{f_2}X_{g_2}Y_5 \end{cases}$$

$$4^{th} \text{ form in Definition 4.14} \begin{cases} T_k(y) &= Y_1X_1X_2X_3X_4Y_5 \\ T_k(y) &= Y_1X_{g_1}X_{f_1}X_{f_2}X_{g_2}Y_5 \end{cases}$$

where, f_1, g_1, f_2 and g_2 is same as defined earlier.

Here all X_i and Y_i have length multiple of k as they form k-tuple representation of y and y'. We can think of $T_k(y)$ and $T_k(y')$ as strings in the alphabet $\Sigma_{y,k}$ formed by k-length factors of y or y'. X_i and Y_i are substrings. Here y and y' takes the form given in Lemma 4.12 and so $OP_1(T_k(y)) = OP_1(T_k(y'))$ when we consider $T_k(y), T_k(y') \in \Sigma^*_{y,k}$. Hence we have $OP_k(y) = OP_k(y')$. □

Let us denote the transitive closure of the relation by $\underset{mTr}{\overset{k+}{\equiv}}$. We say $\mathbb{TR}_k(u) = \{v \mid v \underset{mTr}{\overset{k+}{\equiv}} u\}$. Next few results will give us the minimum length of a string u to have $|\mathbb{TR}_k(u)| \geq 2$.

Lemma 4.16. If $y, y' \in \Sigma$ for some finite alphabet Σ with $y \neq y'$ and y' is obtained from y by applying k-mirror transposition of the first and third form in Definition 4.14 then $|y| = |y'| \geq 3k + 2$.

Proof. Let the length of both the strings is η. So, $|T_{(k-1)}(y)| = |T_{(k-1)}(y')| = (k-1)(\eta-(k-1)+1) = (k-1)(\eta-k+2)$. In case of the first and third form in Lemma 4.14 we have y and y' are in the form

$$T_{(k-1)}(y) = UU' \text{ and } T_{(k-1)}(y) = VV'$$

such that

$$\begin{aligned} m_{[1,k]}(T^{-1}_{(k-1)}(U)) &= m_{[1,k]}(T^{-1}_{(k-1)}(V)) \\ &\text{and} \\ m_{[1,k]}(T^{-1}_{(k-1)}(U')) &= m_{[1,k]}(T^{-1}_{(k-1)}(V')) \end{aligned}$$

where,

(i) $(1^{st} form)$

$$\begin{aligned} U &= y_1z_1y_2z_2y_3z_1y_4z_2, & U' &= y_5z_1y_6z_2y_7z_1y_8z_2y_9 \\ V &= y_1z_1y_{q_1}z_2y_3z_1y_{p_1}z_2, & V' &= y_5z_1y_{p_2}z_2y_7z_1y_{q_2}z_2y_9 \end{aligned}$$

(ii) $(3^{rd} form)$

$$\begin{aligned} U &= y_1zy_2zy_4z, & U' &= y_5zy_6zy_8zy_9 \\ V &= y_1zy_{q_1}zy_{p_1}z, & V' &= y_5zy_{p_2}zy_{q_2}zy_9 \end{aligned}$$

We know from [40] if two strings are k-abelian equivalent then their minimum length is $2k$. So, $U$and V will have minimum length $(k-1)(2k-(k-1)+1) = (k-1)(k+2)$. Similarly U' and V' will have minimum length $(k-1)(k+2)$. So,

$$|T_{(k-1)}(y)| = (k-1)(\eta - k + 2) \geq (k-1)(2k+4)$$

. This means $\eta \geq 3k+2$. □

Lemma 4.17. If $y, y' \in \Sigma$ for some finite alphabet Σ with $y \neq y'$ and y' is obtained from y by applying k-mirror transposition of the second or fourth form in Lemma 4.14 then $|y| = |y'| \geq 3k+1$.

Proof. Let the length of both the strings is η. So, $|T_{(k-1)}(y)| = |T_{(k-1)}(y')| = (k-1)(\eta-(k-1)+1) = (k-1)(\eta-k+2)$. In case of second and fourth form in Lemma

4.14 we have y and y' are in the form

$$T_{(k-1)}(y) = UzU' \text{ and } T_{(k-1)}(y) = VzV'$$

such that $|z| = k-1$ and

$$\begin{aligned} m_{[1,k]}(T^{-1}_{(k-1)}(Uz)) &= m_{[1,k]}(T^{-1}_{(k-1)}(zV)) \\ &\text{and} \\ m_{[1,k]}(T^{-1}_{(k-1)}(U'z)) &= m_{[1,k]}(T^{-1}_{(k-1)}(zV')) \end{aligned}$$

where ,

(i) $(2^{nd} form)$

$$\begin{aligned} U &= y_1zy_2zy_3zy_4, & U' &= y_6zy_7zy_8zy_9 \\ V &= y_1zy_{q_1}zy_3zy_{p_1}, & V' &= y_{p_2}zy_7zy_{q_2}zy_9 \end{aligned}$$

(ii) $(4^{th} form)$

$$\begin{aligned} U &= y_1zy_2zy_4, & U' &= y_6zy_8zy_9 \\ V &= y_1zy_{q_1}zy_{p_1}, & V' &= y_{p_2}zy_{q_2}zy_9 \end{aligned}$$

We know from [40] if the two strings are k-abelian equivalent then their minimum length is $2k$. So, Uz and zV will have minimum length $(k-1)(2k-(k-1)+1)$ $= (k-1)(k+2)$. Similarly $U'z$ and zV' will have minimum length $(k-1)(k+2)$. So,

$$|T_{(k-1)}(y)| = (k-1)(\eta - k + 2) \geq (k-1)(2k+4) - (k-1).$$

This means $\eta \geq 3k+1$. □

Theorem 4.18. If $u \in \Sigma^n$ for some finite alphabet Σ with $|\Sigma| \geq 2$ then $|\mathbb{TR}_k(u)| = 1$ $\forall k \geq \lfloor \frac{n-1}{3} \rfloor + 1$

Proof. If $v \in \mathbb{TR}_k(u)$ and $v \neq u$ then v is obtained from applying atleast one k-mirror transposition. From Lemma 4.16 and Lemma 4.17 we have at least $3k+1$ as the length of the string such applying k-transposition on the string results in different strings. Hence we can have the value of k at most $\lfloor \frac{n-1}{3} \rfloor$ to have two different strings in $\mathbb{TR}_k(u)$. Hence the result follows. □

Although it seems that for a given k the minimal string pairs having k-precedence data same is $3k+1$ but we can find more than two pairs. For example these following pairs of strings with length 31 have the same 10-length precedence data pairwise:

000000000	10	000000000	01	000000000		101010101	10	101010101	01	101010101
000000000	01	000000000	10	000000000		101010101	01	101010101	10	101010101
000010000	10	000010000	01	000010000		101101101	10	101101101	01	101101101
000010000	01	000010000	10	000010000		101101101	01	101101101	10	101101101
001000100	10	001000100	01	001000100		110111011	10	110111011	01	110111011
001000100	01	001000100	10	001000100		110111011	01	110111011	10	110111011
010010010	10	010010010	01	010010010		111101111	10	111101111	01	111101111
010010010	01	010010010	10	010010010		111101111	01	111101111	10	111101111
010101010	10	010101010	01	010101010		111111111	10	111111111	01	111111111
010101010	01	010101010	10	010101010		111111111	01	111111111	10	111111111

Table 4.1: Examples of string pairs (u, v) where $OP_{10}(u) = OP_{10}(v)$ and $|u| = |v| = 31$.

Chapter 5

Separating Words and Discrepancy Theory

5.1 Introduction

The well known separating words problem is to find the smallest DFA to distinguish between two words of length $\leq n$. In this chapter we generalize the idea of separating words by modular factor composition and introduce precedence data on modular factor composition of a word. We show that precedence data on l-modular factor composition can be same for two strings of length $O(l^4 \log n)$. The result is similar to which is obtained using l-modular factor composition introduced in [57]. We also use l-modular composition of words to find asymptotic value of the smallest l such that a l-regular hypergraph has non-zero discrepancy with respect to a partial coloring.

5.2 Separating Words Problem and its Motivation

When two very long input strings are provided, it may be very difficult to distinguish them when they cannot be scanned together. While scanning a single string with very little memory (an amount far less than what is needed to store actual strings) we should look for certain key characteristics of the string that can fit into that amount of memory. Then hope for the second string to not satisfy that set of characteristics

to achieve separation. Given two strings if there is a need to separate them repeatedly one can compute certain characteristics with the help of large memory and based on that a rule is formed such that one string satisfies that rule but the other do not while they are scanned individually. Here the rule separates those given pair of strings. From the point of view of theory of computation those rules can be a grammar, a finite automaton, a Turing machine etc.

Deterministic finite automaton (DFA) has been studied in this regard since 1986 [17]. The famous question under in this context is: given any two strings of length $\leq n$, how many states forming a DFA are sufficient separating them?

A Deterministic Finite Automaton (DFA) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite non-empty set of states, Σ is a non-empty input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of final states. The language accepted by an automaton is the set $\{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$, where $\hat{\delta}$ is the extension of transition function δ over strings. For further details on this topic, we refer to [23].

Let u and v be two distinct finite strings defined over an alphabet Σ. We say that an automaton, M, separates strings u and v if M accepts u but rejects v. Given two distinct strings u and v, let $dsep(u, v)$ be the number of states in the smallest DFA accepting u and rejecting v. Separating words problem is inverse of classical problem wherein we seek smallest string that can distinguish two DFAs say with m and n states, which can be done by a $m + n - 1$ length string [51].

We consider the question of finding smallest finite automaton that separates or distinguish string u from v. This has been studied earlier in the context of data encoding in [27] and independently in [10, 17, 46]. Demaine et. al. [10] and Currie et. al. [8] have also studied separating words by non-deterministic finite automaton and by context free grammar, respectively. In [51], the improvement of upper bound $O(n^{2/5} \log^{3/5} n)$ for separating word by deterministic finite automaton is mentioned as an open problem. Consider for example, the DFA having three states shown in Figure 5.1 that separates 0010 from 0100. It can be shown by considering all possible DFAs having two states, that no DFA with two states can separate these two strings. Observe that $dsep(u, v) = dsep(v, u)$ as the complement of an DFA can be obtained

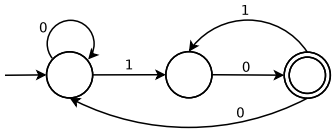

Figure 5.1: DFA that separates 0010 from 0100.

by interchanging the accept and reject states. Note that there may be more than one non-equivalent DFA with the same number of states that separates u and v. We define $S(n) = \max_{\substack{u \neq v \\ |u|,|v| \leq n}} dsep(u, v)$. The separating words problem is to determine upper and lower bounds on $S(n)$ [17]. Let u and v be two strings with $|u|, |v| \leq n$. If $|\Sigma| = 1$ (unary alphabet) or $|u| \neq |v|$ then a DFA with $O(\log n)$ states is necessary and sufficient that accepts u but rejects v, or vice versa [17].

It is shown in [10, 17] that the size of alphabet does not affect $S(n)$. In other words, if $S_k(n)$ is the maximum number of states needed to separate two length-n strings over an alphabet of size k, then $S_k(n) = S_2(n)$ for all $k \geq 2$. Thus, it suffices to consider binary alphabet for analysis. Since the strings of unequal size are easy to separate, similar to [10], we focus on strings of equal length and redefine $S(n) = \max_{\substack{w \neq x \\ |w|,|x|=n}} dsep(u, v)$.

Although $S(n)$ has the best known upper bound $O(n^{\frac{2}{3}} \log^{\frac{3}{5}} n)$, restricting DFA to be reversible automaton gives us $O(\sqrt{n})$ to be the best known upper bound [47] found by Robson. A reversible automaton is a finite (possibly incomplete) automaton in which each letter induces a partial one-to-one map from the set of states into itself [41]. Vyalyi and Gimadeev [57] generalized the idea and used modular factor composition of words and their scheme gives a lower bound on l to be $\Omega(n^{\frac{1}{3}} \log^{\frac{-1}{3}} n)$ needed to have different l-modular factor composition between any pair of n-length strings. In the next section we shall discuss modular factor composition of a word in a broader perspective. In Section 5.4 we shall introduce precedence data in modular factor composition.

5.3 Modular Factor Composition of a Word

In this section we shall modify the definition of l-modular factor composition of a word and relate it to k-abelian equivalence of words and use similar techniques as used in [57] to find the minimal length of a pair of strings having distinct l-modular factor composition. The l-modular factor composition used in [57] is a subset of the l-modular factor composition we shall define in this section.

Let $s \in \Sigma^n$ for some finite alphabet Σ with $|\Sigma| \geq 2$. The set F_k^s is the lexicographically ordered set of distinct k-length factors of s. The set ${}_a^d F_k^s$ is the subset of F_k^s and is the set of all distinct k length factors of s appearing at positions congruent to a mod d. ${}_a^d F_k^s$ is also lexicographically ordered. The k-length factor multiset of s at positions with index congruent to a mod d is expressed by the vector $[{}_a^d \mathcal{M}_k^s]$ where $[{}_a^d \mathcal{M}_k^s](i)$ is the frequency of ${}_a^d F_k^s(i)$.

Definition 5.1 (l-modular k-factor composition)**.** For $s \in \Sigma^n$ with $|\Sigma| \geq 2$, then the ordered pair

$$\mathrm{mod}_k^l(s) = \{([{}_a^d \mathcal{M}_k^s], {}_a^d F_k^s) \mid 0 \leq a < d \leq l\}$$

is said to be l-modular k-factor composition of the string s.

Lemma 5.2. If $\mathrm{mod}_l^l(v) = \mathrm{mod}_l^l(w)$ then $\mathrm{mod}_i^l(v) = \mathrm{mod}_i^l(w)$ $\forall 1 \leq i < l$ if and only if w and v are l-abelian equivalent words.

Proof. Let w and v be l-abelian equivalent words. Then $\mathrm{suff}_{l-1}(v) = \mathrm{suff}_{l-1}(w)$.

Let u be any string of length h such that $1 \leq h < l$. Frequency of u in v as a factor at positions congruent a mod d is denoted by ${}_a^d|v|_u$.

Let $\{U_1, \ldots, U_t\}$ be the set of distinct l-length factors of v such that u is a prefix of every U_i $\forall i \in \{1, \ldots, t\}$. Then we have,

$${}_a^d|v|_v = \sum_{i=1}^{t} {}_a^d|v|_{U_i} + |v_j v_{j+1} \ldots v_{j+h-1}|_u \text{ where, } j = n - ((n - a + 1) \bmod d) + 1$$

As $([{}^{d}_{a}\mathcal{M}^{v}_{l}], {}^{d}_{a}F^{v}_{l}) = ([{}^{d}_{a}\mathcal{M}^{w}_{l}], {}^{d}_{a}F^{w}_{l})$ we have

$${}^{d}_{a}|w|_{u} = \sum_{i=1}^{t} {}^{d}_{a}|w|_{U_i} + |w_j w_{j+1} \dots w_{j+h-1}|_{u} \text{ where, } j = n - ((n-a+1) \bmod d) + 1$$

The smallest value of j is $n-l+2$ when $d = l$. As, $\text{suff}_{l-1}(v) = \text{suff}_{l-1}(w)$, we have $|v_j v_{j+1} \dots v_{j+h-1}|_{u} = |w_j w_{j+1} \dots w_{j+h-1}|$. So, ${}^{d}_{a}|v|_{v} = {}^{d}_{a}|w|_{u}$. Hence we have one side of the proof.

Conversely, if $\text{mod}^{l}_{l}(v) = \text{mod}^{l}_{l}(w)$ and $\text{mod}^{l}_{i}(v) = \text{mod}^{l}_{i}(w)$ $\forall 1 \leq i < l$ then we have $([{}^{1}_{0}\mathcal{M}^{v}_{l}], {}^{1}_{0}F^{v}_{l}) = ([{}^{1}_{0}\mathcal{M}^{w}_{l}], {}^{1}_{0}F^{w}_{l})$ and $([{}^{1}_{0}\mathcal{M}^{v}_{l-1}], {}^{1}_{0}F^{v}_{l-1}) = ([{}^{1}_{0}\mathcal{M}^{w}_{l-1}], {}^{1}_{0}F^{w}_{l-1})$ which means v and w are l-abelian equivalent words. □

Corollary 5.3. If $\text{mod}^{l}_{l}(v) = \text{mod}^{l}_{l}(w)$ and $\text{mod}^{l}_{l-1}(v) = \text{mod}^{l}_{l-1}(w)$ then $\text{mod}^{l}_{i}(v) = \text{mod}^{l}_{i}(w)$ $\forall 1 \leq i \leq l$

Definition 5.4 (l-modular factor composition)**.** If $u \in \Sigma^n$ with $|\Sigma| \geq 2$ then l-modular factor composition of u is defined as:

$$\text{mod}_{l}(u) = \text{mod}^{l}_{l}(u) \cup \text{mod}^{l}_{l-1}(u)$$

We say the two words u and v are l-modular abelian equivalent if and only if $\text{mod}_{l}(u) = \text{mod}_{l}(v)$

Theorem 5.5. The lower bound of l such that any two strings $u, v \in \Sigma^n$ are not l-modular abelian equivalent words is $\Omega(n^{\frac{1}{3}} \log^{\frac{-1}{3}} n)$.

Proof. We shall prove the result by showing the existence of a pair of binary strings of length $O(l^3 \log l)$ being l-modular abelian equivalent.

Consider strings $\in \{0,1\}^m$ such that every pair of string w, s will give us $\text{mod}^{l}_{1}(w) \neq \text{mod}^{l}_{1}(s)$. Expressing $\text{mod}^{l}_{1}(w)$ for every w requires at most $(\frac{l(l+1)}{2} \log m)$ bits as there are $\frac{l(l+1)}{2}$ possible choices of a, d with $0 \leq a < d \leq l$ and for each of them the frequency of number of '1's in the binary string w we may require at most $\log m$ bits. Those bits can express $2^{\frac{l(l+1)}{2} \log m}$ different configurations. Also there are 2^m different binary strings. We need to attach an unique configuration to each such

string. In order to achieve that we need $\frac{l(l+1)}{2}\log m \geq m$ or $l = \Omega(m^{\frac{1}{2}}\log^{\frac{-1}{2}} m)$. If $m = \frac{l(l+1)}{2}\log m + 1$ then we have a pair of strings $w, s \in \{0,1\}^m$ such that $\mathrm{mod}_l(w) = \mathrm{mod}_l(s)$. Here $m = O(l^2 \log l)$.

We shall have a prime number p such that $l < p < 2l$ by Bertrand-Chebyshev Theorem. Consider the operation on any binary string of length m,

$$\tau_p(u)_i = \begin{cases} \text{'1'}, & \text{iff } i = pj \text{ for some } j \in \mathbb{N} \text{ and } u_j = \text{'1'} \\ \text{'0'}, & \text{otherwise} \end{cases}$$

The operation τ_p results in another binary string $\tau_p(u)$ of length $pm + p - 1$ and the letter with index i in $\tau_p(u)$ is denoted as $\tau_p(u)_i$. Choose a, d such that $0 \leq a < d \leq l$. The possible factors of length l of $\tau_p(w)$ are:

$$\{0^l\} \cup \left(\bigcup_{q=0}^{l-1} \{0^q 10^{l-q-1}\} \right)$$

$0^q 10^{l-q-1}$ is at position starting with index t in $\tau_p(w)$

$\Leftrightarrow$ 1 is at position with index $t + q$ in $\tau_p(w)$

So, $0^q 10^{l-q-1}$ is at position starting with index $a + di$ in $\tau_p(w)$

$\Leftrightarrow$ 1 is at position with index $a + q + di$ in $\tau_p(w)$

$\Leftrightarrow$ 1 is at position with index $\dfrac{a + q + di}{p}$ in w

$\Leftrightarrow$ 1 is at position with index congruent to $(a + q)p^{-1} \bmod d$ in w

Solving for x in linear congruence $px = (a + q) \bmod d$,

here p^{-1} is the modular inverse of $p \bmod d$

$\Leftrightarrow$ 1 is at position with index congruent to $(a + q)p^{-1} \bmod d$ in s

$\Leftrightarrow$ 1 is at position with index $\dfrac{a + q + di'}{p}$ in s

$\Leftrightarrow$ $0^q 10^{l-q-1}$ is at position starting with index $a + di'$ in $\tau_p(s)$.

Hence, ${}_a^d|\tau_p(w)|_{0^q 10^{l-q-1}} = {}_a^d|\tau_p(s)|_{0^q 10^{l-q-1}}$ $\forall q \in \{0, 1, \ldots, l-1\}$ and it also ensures

${}^{d}_{a}|\tau_p(w)|_{0^l} = {}^{d}_{a}|\tau_p(s)|_{0^l}$

We have, $\mathrm{mod}^l_l(\tau_p(w)) = \mathrm{mod}^l_l(\tau_p(s))$.

Now, ${}^{d}_{a}|\tau_p(w)|_{0^q10^{l-q-2}} = {}^{d}_{a}|\tau_p(w)|_{0^q10^{l-q-1}} = {}^{d}_{a}|\tau_p(w)|_{0^q10^{l-q-1}} = {}^{d}_{a}|\tau_p(w)|_{0^q10^{l-q-2}}$ $\forall q \in \{0, 1, \ldots, l-2\}$.

Also, ${}^{d}_{a}|\tau_p(w)|_{0^{l-1}} = {}^{d}_{a}|\tau_p(w)|_{0^l} + {}^{d}_{a}|\tau_p(w)|_{0^{l-1}1} = {}^{d}_{a}|\tau_p(s)|_{0^l} + {}^{d}_{a}|\tau_p(s)|_{0^{l-1}1} = {}^{d}_{a}|\tau_p(s)|_{0^{l-1}}$.

Hence, $\mathrm{mod}^l_{l-1}(\tau_p(w)) = \mathrm{mod}^l_{l-1}(\tau_p(s))$ and so we have $\mathrm{mod}_l(\tau_p(w)) = \mathrm{mod}_l(\tau_p(s))$ with $|\tau_p(w)| = |\tau_p(s)| = O(l^3 \log l)$. □

Theorem 5.6. If two words $u, v \in \Sigma^n$, are not l-modular abelian equivalent, then they can be separated by a DFA with $O(l \log n)$ states.

Proof. The trivial cases are:

(i) ${}^{l}_{a}|u|_w \neq {}^{l}_{a}|v|_w$ for some factor w with $|w| = l$ or $|w| = l-1$

(ii) ${}^{l-1}_{a}|u|_w \neq {}^{l-1}_{a}|v|_w$ for some factor w with $|w| = l-1$

These two cases are already discussed in [57] except the scenario where ${}^{l}_{a}|u|_w \neq {}^{l}_{a}|v|_w$ and $|w| = l-1$, which can be taken care of using same idea of construction of DFA used in [57].

Non trivial case: ${}^{d}_{a}|u|_w \neq {}^{d}_{a}|v|_w$ with $|w| = l$ or $|w| = l-1$ but $|w| > d$. Also $0 \leq a < d \leq l-1$. The inequality ${}^{d}_{a}|u|_w \neq {}^{d}_{a}|v|_w$ can be captured by $0 \leq c < b = O(\log n)$ with only one of the sides of the inequality is congruent to c mod b. Let $|w| = k$. The following DFA will accepts only those strings which have w as a factor at positions congruent to a mod d with frequency congruent to c mod b.
Let the smallest period of w is p and $lcm(d, p) = g$.
Consider the DFA with states:

- $q(i, j)$ for $i \in \{0, 1, \ldots, b-1\}, j \in \{0, 1, 2, \ldots, d-1\}$
- $s(i, j)$ for $i \in \{0, 1, \ldots, b-1\}, j \in \{1, \ldots, k\}$

So, the number of states is $(d + k)b = O(l \log n)$. The starting state is $q(0, 0)$ and the final states are

- $q(c, j)$ $\forall$ $j \in \{0, 1, \ldots, d-1\}$

- $s(c, j)\ \forall\ j \in \{1, \ldots, k-1\}$
- $s((c+1) \bmod b, k)$

The transition function δ is defined by the following equations:

$$\delta(q(i,j),\sigma) = \begin{cases} q(i,(j+1) \bmod d), & \text{if } (a-1) \neq \bmod d \\ s(i,1), & \text{if } u_1 = \sigma \text{ and } j = (a-1) \bmod d \\ q(i,a), & \text{if } u_1 \neq \sigma \text{ and } j = (a-1) \bmod d \end{cases} \tag{5.1}$$

Let $r = j - (j \bmod d) + 1$ and $t = (j \bmod d) + 1$

$$\delta(s(i,j),\sigma) = \left\{ \begin{array}{ll} s(i,j+1), & \text{if } u_{j+1} = \sigma \\ q(i,(j+a) \bmod d, & \text{if } u_{j+1} \neq \sigma \text{ and} \\ & u_r u_{r+1} \ldots u_j \cdot \sigma \neq \text{pref}_t(u) \\ s(i,t) \bmod d, & \text{if } u_{j+1} \neq \sigma \text{ and} \\ & u_r u_{r+1} \ldots u_j \cdot \sigma = \text{pref}_t(u) \end{array} \right\} \forall j \in \{1,2,\ldots,k-1\} \tag{5.2}$$

$$\delta(s(i,k),\sigma) = \begin{cases} q\big((i+1) \bmod b, (k+a) \bmod d\big), & \text{if } g > k \\ s\big((i+1) \bmod b, k-g+1\big), & \text{if } g < k \text{ and } \sigma = u_{k-g+1} \\ q\big((i+1) \bmod b, (k+a) \bmod d\big), & \text{if } g < k \text{ and } \sigma \neq u_{k-g+1} \\ s\big((i+1) \bmod b, 1\big), & \text{if } g = k \text{ and } u_1 = \sigma \\ q\big((i+1) \bmod b, (k+a) \bmod d\big), & \text{if } g = k \text{ and } u_1 \neq \sigma \end{cases} \tag{5.3}$$

The DFA consists of b levels (denoted by first component of the states) and each level has two parts ($q(i,j)$ and $s(i,j)$). We call them q-part and s-part respectively. So, each level has $d+k$ states. When the DFA encounters an occurrence of u the transition function takes the execution of the DFA to the next level modulo b.

Equation 5.1 ensures that for each level the execution of the DFA goes from q-part to the s-part only when it encounters the first letter of u from the input string at positions congruent to a mod d. The other cases in Equation 5.1 ensures that the execution of DFA moves in a cyclic way and stays in the q-part.

Equation 5.2 ensures after reading one occurrence of u in the string at some position congruent to a mod d, the execution of DFA remains at states $s(i,k)$ for every level related to i. Equation 5.2 also makes the transition function to take care of the possible self overlapping nature of u and due to that the possibility of encountering the prefix of u at some position congruent to a mod d. Equation 5.2 also ensures that the execution of the DFA moves back to q part in each level whenever u is not read or no self overlapping is encountered at a position congruent to a mod d.

Equation 5.3 handles the transition to from one level to the next level modulo b from the states $s(i,k)$. Reaching at states $s(i,k)$, the execution ensures an occurrence of u in the string at a position congruent to a mod d. The next occurrence of u at a position congruent to a mod d may not need the transition function to start from q-part of the next level due to self overlap and small period of u. The transition function is made to take care of this scenario by Equation 5.3. □

Example 5.7. Let u = 1011 1011 1011 1011 10. Clearly smallest period of u is $p = 4$. Let $d = 3, a = 2$ and $c = 0, b = 3$. Then in the Figure 5.2 we have the DFA accepting the strings with u occurring λ times as a factor at positions congruent to a mod d where λ is congruent to c mod b.

5.4 Precedence data in Modular factor composition

Let $s \in \Sigma^n$ for some finite alphabet Σ with $|\Sigma| \geq 2$. The number of ways a k-length factor appears as a factor at positions congruent to a mod d before another k-length factor of s at positions congruent to b mod d is expressed by the matrix $[{}^{d}_{(a,b)}\mathcal{OP}^{s}_{k}]$ where $[{}^{d}_{(a,b)}\mathcal{OP}^{s}_{k}]_{ij}$ denotes the number of ways the string ${}^{d}_{a}F^{s}_{k}(i)$ appears

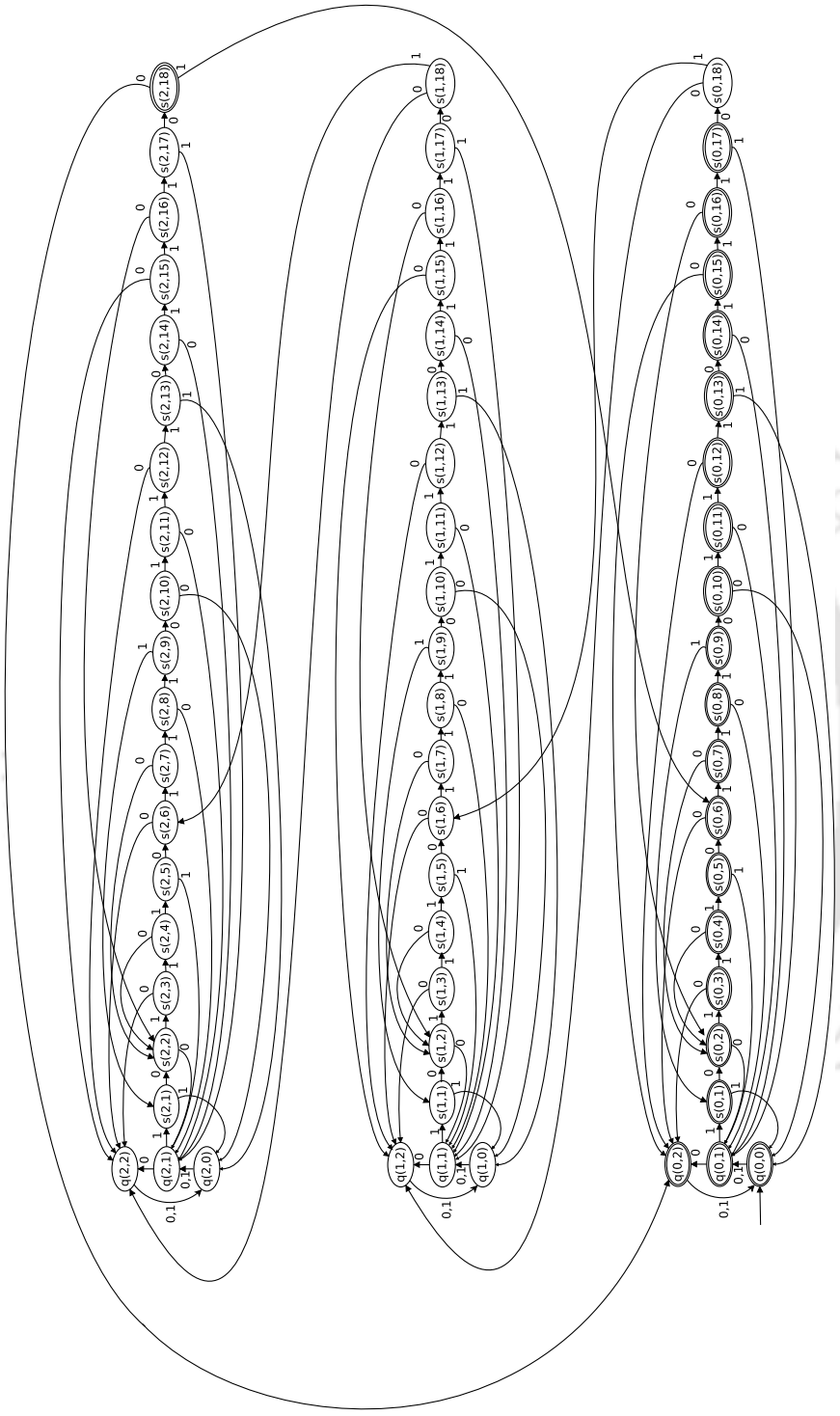

Figure 5.2: DFA accepting strings which has 1011 1011 1011 1011 10 occuring 0 mod 3 times as a factor at positions 2 mod 3.

before ${}_b^d F_k^s(j)$ as a factor in s.

Definition 5.8 (l-modular k-factor precedence composition)**.** For any string $s \in \Sigma^n$ with $|\Sigma| \geq 2$ the collection of ordered triplet:

$$\text{premod}_k^l(s) = \{([{}_{(a,b)}^d \mathcal{OP}_k^s], {}_a^d F_k^s, {}_b^d F_k^s) \mid 0 \leq a < d \leq l, 0 \leq b < d \leq l\}$$

is said to be the l-modular k-factor precedence composition of the string s.

Theorem 5.9. For any two strings v and w,

$$\text{premod}_l^l(v) = \text{premod}_l^l(w) \Rightarrow \text{premod}_k^l(v) = \text{premod}_k^l(w)\ \forall 1 \leq k < l$$

Proof. From the given conditions we have

$$\begin{aligned} ([{}_{(0,0)}^1 \mathcal{OP}_l^v], {}_0^1 F_l^v, {}_0^1 F_l^v) &= ([{}_{(0,0)}^1 \mathcal{OP}_l^w], {}_0^1 F_l^w, {}_0^1 F_l^w) \\ \Rightarrow OP_l(v) &= OP_l(w) \\ \Rightarrow m_{[1,l]}(v) &= m_{[1,l]}(w) \\ \Rightarrow \text{suff}_{l-1}(v) &= \text{suff}_{l-1}(w) \end{aligned}$$

If $x, y \in \Sigma^k$ then the number of ways x can appear as a factor in v at a position congruent to a mod d before y as a factor in v at position congruent to b mod d is given by ${}_{(a,b)}^d|v|_{(x,y)}$. Let $\{X_1, \ldots, X_t\}$ be the set of distinct l-length factors of v such that x is prefix of $X_i\ \forall i \in \{1, \ldots, t\}$ and $\{Y_1, \ldots, Y_r\}$ be the set of distinct l-length factors of v such that y is prefix of $Y_i\ \forall i \in \{1, \ldots, r\}$. Now, let $1 \leq k < l$ then

$${}_{(a,b)}^d|v|_{(x,y)} = \left(\sum_{\substack{i \in \{1,2,\ldots,t\} \\ j \in \{1,2,\ldots,r\}}} {}_{(a,b)}^d|v|_{(X_i,Y_j)} \right) + {}_{(\gamma(a),\gamma(b))}^d|v'|_{(x,y)} + \left({}_a^d|v''|_x \times {}_{\gamma(b)}^d|v'|_y \right)$$

where $\alpha = n - l + 2$, γ is defined as $\gamma(a) = (d - ((\alpha - 1 - a) \bmod d)) \bmod d$ and

$$v' = v_\alpha v_{\alpha+1} \dots v_n$$
$$v'' = v_1 v_2 \dots v_{\alpha+k-2}$$

As $m_{[1,l]}(v) = m_{[1,l]}(w)$, $\{X_1, \dots, X_t\}$ is also the set of distinct l-length factors of w such that x is prefix of X_i $\forall i \in \{1, \dots, t\}$ and $\{Y_1, \dots, Y_r\}$ be the set of distinct l-length factors of w such that y is prefix of Y_i $\forall i \in \{1, \dots, r\}$. So we have,

$${}^{d}_{(a,b)}|w|_{(x,y)} = \left(\sum_{\substack{i \in \{1,2,\dots,t\} \\ j \in \{1,2,\dots,r\}}} {}^{d}_{(a,b)}|w|_{(X_i,Y_j)} \right) + {}^{d}_{(\gamma(a),\gamma(b))}|w'|_{(x,y)} + \left({}^{d}_{a}|w''|_x \times {}^{d}_{\gamma(b)}|w'|_y \right)$$

where

$$w' = w_\alpha w_{\alpha+1} \dots w_n$$
$$w'' = w_1 v_2 \dots w_{\alpha+k-2}$$

As $([{}^{d}_{(a,b)}\mathcal{OP}^v_l], F^v_l) = ([{}^{d}_{(a,b)}\mathcal{OP}^w_l], F^w_l)$ we have

$${}^{d}_{(a,b)}|w|_{(X_i,Y_j)} = {}^{d}_{(a,b)}|w|_{(X_i,Y_j)} \quad \begin{cases} \forall i \in \{1, 2, \dots, t\} \\ \forall j \in \{1, 2, \dots, r\} \end{cases} \tag{5.4}$$

Also as $\text{suff}_{l-1}(v) = \text{suff}_{l-1}(w)$, we have $v' = w'$. We also have

$$([{}^{d}_{(a,b)}\mathcal{OP}^v_l], {}^{d}_{(a,a)}F^v_l) = ([{}^{d}_{(a,b)}\mathcal{OP}^w_l], {}^{d}_{(a,a)}F^w_l)$$

$$\Rightarrow {}^{d}_{a}|v''|_x = {}^{d}_{a}|w''|_x$$

Hence we have the proof. □

Definition 5.10 (l-modular factor precedence data)**.** If $u \in \Sigma^n$ then l-modular factor precedence data of u is $\text{premod}^l_l(u)$. We denote it by $\text{premod}_l(u)$.

We say two words u and v are l-modular factor precedence equivalent if and only if $\text{premod}_l(u) = \text{premod}_l(v)$

Theorem 5.11. The least value of l such that for any two strings $u, v \in \Sigma^n$ are not l-modular factor precedence equivalent is $\Omega(n^{\frac{1}{4}} \log^{\frac{-1}{4}} n)$

Proof. We shall prove the result by showing existence of a pair of binary strings w and s such that their lengths is $O(l^4 \log l)$ and they are l-modular factor precedence equivalent. Consider strings in $\{0,1\}^m$ such that every pair of string u and v will give $\text{premod}_1^l(u) \neq \text{premod}_1^l$. Expressing $\text{premod}_1^l(u)$ require at most $\frac{l(l+1)(2l+1)}{6} \log m$ bits. Following the same argument as in Theorem 5.5 we have $m = O(l^3 \log l)$ such that $\exists u, v \in \Sigma^m$ with $\text{premod}_1^l(u) = \text{premod}_1^l$. Also in a similar manner as Theorem 5.5 we shall choose a prime p such that $l < p < 2p$ we shall show that $\text{premod}_l(\tau_p(u)) = \text{premod}_l(\tau_p(v))$. Hence we shall prove the desired result.

We know the possible l-lenth factors of $\tau_p(u)$ are given by:

$$\{0^l\} \cup \left(\bigcup_{q=0}^{l-1} \{0^q 10^{l-q-1}\} \right)$$

For any $0 \leq q, r < l$ we have:

$$\begin{aligned} {}^{d}_{(a,b)}|\tau_p(u)|_{(0^q10^{l-q-1},0^r10^{l-r-1})} &= {}^{d}_{(a+q,b+r)}|\tau_p(u)|_{(1,1)} = {}^{d}_{(a',b')}|u|_{(1,1)} \text{ and} \\ {}^{d}_{(a,b)}|\tau_p(v)|_{(0^q10^{l-q-1},0^r10^{l-r-1})} &= {}^{d}_{(a+q,b+r)}|\tau_p(v)|_{(1,1)} = {}^{d}_{(a',b')}|v|_{(1,1)} \end{aligned}$$

where, $a' = ((a+q)p^{-1}) \bmod d$ and $b' = ((a+q)p^{-1}) \bmod d$ with p^{-1} is the modular inverse of $p \bmod d$.

The condition $\text{premod}_1^l(u) = \text{premod}_1^l(v)$ means ${}^{d}_{(a',b')}|u|_{(1,1)} = {}^{d}_{(a',b')}|v|_{(1,1)}$ which ensures that ${}^{d}_{(a,b)}|\tau_p(u)|_{(0^q10^{l-q-1},0^r10^{l-r-1})} = {}^{d}_{(a,b)}|\tau_p(v)|_{(0^q10^{l-q-1},0^r10^{l-r-1})}$.

Number of 0^l's in $\tau_p(u)$ at positions congruent to a mod d is given by

$$\begin{aligned}{}_a^d|\tau_p(u)|_{0^l} = &\left(\sum_{i=0}^{d-1} {}_i^d|u|_1.f(a,i,p,l,d)\right) + \left(\sum_{i=0}^{d-1} {}_i^d|u|_0.f(a,i,p,l,d)\right) + \left(\sum_{i=0}^{d-1} {}_i^d|u|_0.g(a,i,p,l,d)\right) \\ &+ \left(\left\lfloor \frac{p|u|+p-l-a}{d} \right\rfloor - \left\lfloor \frac{p|u|+1-a}{d} \right\rfloor\right)\end{aligned}$$

Here,

$$\left.\begin{aligned} f(a,i,p,l,d) &= \left\lceil \tfrac{p-t-l}{d} \right\rceil \\ g(a,i,p,l,d) &= \begin{cases} \left\lceil \frac{p-t}{d} \right\rceil & \text{if } l \geq p-t \\ \left\lceil \frac{l-|i-a|}{d} \right\rceil & \text{otherwise} \end{cases} \end{aligned}\right\} \text{ where, } t = (a-i+p-1) \bmod d$$

For any $0 \leq q < l$ we have:

$$\begin{aligned}{}_{(a,b)}^d|\tau_p(u)|_{(0^l,0^q10^{l-q-1})} = &\left(\sum_{i=0}^{d-1} {}_{(i,b')}^d|u|_{(1,1)}.f(a,i,p,l,d)\right) + \left(\sum_{i=0}^{d-1} {}_{(i,b')}^d|u|_{(0,1)}.f(a,i,p,l,d)\right) \\ &+ \left(\sum_{i=0}^{d-1} {}_{(i,b')}^d|u|_{(0,1)}.g(a,i,p,l,d)\right)\end{aligned}$$

Similarly,

$$\begin{aligned}{}_{(a,b)}^d|\tau_p(v)|_{(0^l,0^q10^{l-q-1})} = &\left(\sum_{i=0}^{d-1} {}_{(i,b')}^d|v|_{(1,1)}.f(a,i,p,l,d)\right) + \left(\sum_{i=0}^{d-1} {}_{(i,b')}^d|v|_{(0,1)}.f(a,i,p,l,d)\right) \\ &+ \left(\sum_{i=0}^{d-1} {}_{(i,b')}^d|v|_{(0,1)}.g(a,i,p,l,d)\right)\end{aligned}$$

Also,

$$
\begin{aligned}
{}^{d}_{(a,b)}|\tau_p(u)|_{(0^q10^{l-q-1},0^l)} = &\left(\sum_{i=0}^{d-1} {}^{d}_{(a',i)}|u|_{(1,1)}.f(b,i,p,l,d)\right) + \left(\sum_{i=0}^{d-1} {}^{d}_{(a',i)}|u|_{(1,0)}.f(b,i,p,l,d)\right) \\
&+ \left(\sum_{i=0}^{d-1} {}^{d}_{(i,b')}|u|_{(1,0)}.g(b,i,p,l,d)\right) \\
&+ {}^{d}_{a}|u|_1 \left(\left\lfloor \frac{p|u|+p-l-b}{d} \right\rfloor - \left\lfloor \frac{p|u|+1-b}{d} \right\rfloor\right)
\end{aligned}
$$

and,

$$
\begin{aligned}
{}^{d}_{(a,b)}|\tau_p(v)|_{(0^q10^{l-q-1},0^l)} = &\left(\sum_{i=0}^{d-1} {}^{d}_{(a',i)}|v|_{(1,1)}.f(b,i,p,l,d)\right) + \left(\sum_{i=0}^{d-1} {}^{d}_{(a',i)}|v|_{(1,0)}.f(b,i,p,l,d)\right) \\
&+ \left(\sum_{i=0}^{d-1} {}^{d}_{(i,b')}|v|_{(1,0)}.g(b,i,p,l,d)\right) \\
&+ {}^{d}_{a}|v|_1 \left(\left\lfloor \frac{p|u|+p-l-b}{d} \right\rfloor - \left\lfloor \frac{p|u|+1-b}{d} \right\rfloor\right)
\end{aligned}
$$

$\text{premod}^l_1(u) = \text{premod}^l_1(v)$ ensures that ${}^{d}_{(a,b)}|\tau_p(u)|_{(0^l,0^q10^{l-q-1})} = {}^{d}_{(a,b)}|\tau_p(v)|_{(0^l,0^q10^{l-q-1})}$ and ${}^{d}_{(a,b)}|\tau_p(u)|_{(0^q10^{l-q-1},0^l)} = {}^{d}_{(a,b)}|\tau_p(v)|_{(0^q10^{l-q-1},0^l)}$

Let $U \cdot 0^p = \tau_p(u)$ and $V \cdot 0^p = \tau_p(u)$.
So, ${}^{d}_{(a,b)}|\tau_p(u)|_{(0^l,0^l)} = {}^{d}_{(a,b)}|\tau_p(U)|_{(0^l,0^l)} + {}^{d}_{(\tilde{a},\tilde{b})}|0^p|_{(0^l,0^l)} + {}^{d}_{a}|\tau_p(U)|_{0^l} \times {}^{d}_{\tilde{b}}|o^p|_{0^l}$
and similarly, ${}^{d}_{(a,b)}|\tau_p(v)|_{(0^l,0^l)} = {}^{d}_{(a,b)}|\tau_p(V)|_{(0^l,0^l)} + {}^{d}_{(\tilde{a},\tilde{b})}|0^p|_{(0^l,0^l)} + {}^{d}_{a}|\tau_p(V)|_{0^l} \times {}^{d}_{\tilde{b}}|o^p|_{0^l}$,
where $\tilde{a} = (d - (p|u|) \bmod d + a) \bmod d$ and $\tilde{b} = (d - (p|u|) \bmod d + b) \bmod d$.

We can readily see that ${}^{d}_{(a,b)}|\tau_p(u)|_{(0^l,0^l)} = {}^{d}_{(a,b)}|\tau_p(v)|_{(0^l,0^l)}$ if and only if ${}^{d}_{(a,b)}|\tau_p(U)|_{(0^l,0^l)} = {}^{d}_{(a,b)}|\tau_p(V)|_{(0^l,0^l)}$.

Now,

$$
\begin{aligned}
{}^{d}_{(a,b)}|\tau_p(U)|_{(0^l,0^l)} = &\left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} {}^{d}_{(i,j)}|U|_{(1,1)}.f(a,i,p,l,d).f(b,i,p,l,d)\right)\\
&+\left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} {}^{d}_{(i,j)}|U|_{(1,0)}.f(a,i,p,l,d).f(b,i,p,l,d)\right)\\
&+\left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} {}^{d}_{(i,j)}|U|_{(1,0)}.f(a,i,p,l,d).g(b,i,p,l,d)\right)\\
&+\left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} {}^{d}_{(i,j)}|U|_{(0,1)}.f(a,i,p,l,d).f(b,i,p,l,d)\right)\\
&+\left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} {}^{d}_{(i,j)}|U|_{(0,0)}.f(a,i,p,l,d).f(b,i,p,l,d)\right)\\
&+\left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} {}^{d}_{(i,j)}|U|_{(0,0)}.f(a,i,p,l,d).g(b,i,p,l,d)\right)\\
&+\left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} {}^{d}_{(i,j)}|U|_{(0,1)}.g(a,i,p,l,d).f(b,i,p,l,d)\right)\\
&+\left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} {}^{d}_{(i,j)}|U|_{(0,0)}.g(a,i,p,l,d).f(b,i,p,l,d)\right)\\
&+\left(\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} {}^{d}_{(i,j)}|U|_{(0,0)}.g(a,i,p,l,d).g(b,i,p,l,d)\right)
\end{aligned}
$$

Similarly, ${}^{d}_{(a,b)}|\tau_p(V)|_{(0^l,0^l)}$ can be expressed by substituting U by V in the previous expression. Which again by using $\text{premod}^l_1(u) = \text{premod}^l_1(v)$, proves ${}^{d}_{(a,b)}|\tau_p(u)|_{(0^l,0^l)} = {}^{d}_{(a,b)}|\tau_p(v)|_{(0^l,0^l)}$.

Hence, we have shown that $\exists\ w = \tau_p(u)$ and $s = \tau_p(v)$ such that $w, s \in \{0,1\}^n$ with $n = O(l^{\frac{1}{4}} \log l)$ and $\text{premod}_l(w) = \text{premod}_l(s)$. □

5.4.1 Remarks

We have not found any upper bound on l in terms of n such that for any two strings u and v of length n we shall have $\text{premod}_l(u) \neq \text{premod}_l(u)$. An obvious upper bound is Robson's upper bound of $O(\sqrt{n})$. Any improvement in the upper bound may give us the improvement on the problem of Separating Words with DFA. Apart from the relation with Separating words problem the structures $\text{mod}_l(u)$ and $\text{premod}_l(u)$ introduced by us have interesting properties related to k abelian equivalence of words. They behave in quite similar way as $m_{[1,k]}(u)$ and $OP_{[1,k]}(u)$ respectively.

5.5 Discrepancy of Hypergraph for Partial Coloring

5.5.1 Definitions and Notations

Let (V, S) be a set system where $V = \{v_1, v_2, ..., v_n\}$ be a set of points and $S = \{S_1, S_2, ..., S_m\}$ be a collections of sets and $S_i \subseteq V \ \forall i \in \{1, 2, ..., m\}$. We shall abuse the notation and use v as $\{1, 2, ..., n\}$.

This set system gives us a hyper-graph with V as vertex-set and S as set of hyper-edges. We shall use (V, S) to mean both a set system and the corresponding hyper-graph.

Definition 5.12 (k-uniform hypergraph)**.** A hypergraph (V, S) is called k-uniform if $|S_i| = k \ \forall i \in \{1, 2, ..., m\}$.

Definition 5.13 (k-regular hypergraph)**.** A hypergraph (V, S) is called k-regular if every vertex appears exactly at k hyper-edges.
i.e if for a hypergraph (V, S)

$$\delta(i, j) = \begin{cases} 1 \text{ if } v_i \in S_j \\ 0 \text{ otherwise,} \end{cases}$$

and $\sum_{j=1}^{m} \delta(i,j) = k \ \forall i \in \{1,2,\ldots,n\}$ then it is a k-regular hypergraph.

Let $\chi : \{v_1, v_2, .., v_n\} \to \{-1, 0, 1\}$ be function which can be viewed as partial 2-coloring of vertices of the hypergraph (V, S) if χ does not map to only 0. $\mathcal{C}$ be the set of all partial colorings on $\{v_1, v_2, .., v_n\}$. A function $\hat{\chi} : \{v_1, v_2, .., v_n\} \to \{-1, 1\}$ can be viewed as total coloring of vertices of the hypergraph (V, S). Let $\mathcal{C}_T$ be the set of all total colorings on $\{v_1, v_2, .., v_n\}$. We can easily see that $\mathcal{C}_T \subset \mathcal{C}$.

For each subset S_i of S, ideally not every v_j will be in S_i. So $\sum_{v_j \in S_i} \chi(v_j)$ gives us the imbalance of partial coloring in each S_i. This imbalance can be both in positive and negative direction depending on either 1 or -1 is more in number respectively. The value of this imbalance can be given as $|\sum_{v_j \in S_i} \chi(v_j)|$.

For a given set system S, $\max_{S_i \in S} |\sum_{v_j \in S_i} \chi(v_j)|$ gives the maximum possible imbalance of a set system with given partial color.

We define discrepancy of a hypergraph (V, S) with respect to a set of partial colorings $P \subseteq \mathcal{C}$ by

$$D_P(V, S) = \min_{\chi \in P} \max_{S_i \in S} |\sum_{v_j \in S_i} \chi(v_j)|.$$

We shall use $D_P(S)$ to denote the discrepancy of the set system (V, S) where $V = \{1, 2, ..., n\}$. We say that the set system S on $\{1, 2, ..., n\}$ have a perfect partial coloring if $D_P(S) = 0$ i.e. there exists a partial 2-coloring χ such that the number of colors in each S_i is equal. The following theorem gives us the necessary size of the set system to have a perfect partial coloring in $\{1, 2, ..., n\}$.

Theorem 5.14. [54] If $\mathcal{S}$ be any set system on $\{1, 2, \ldots, n\}$ and if

$$\prod_{s \in \mathcal{S}} (1 + |s|) \leq 2^n$$

then $\mathcal{S}$ always have a perfect partial coloring.

Non existence of perfect partial coloring on $\{1, 2, ..., n\}$ is equivalent to having $D_{\mathcal{C}}(S) \neq 0$. The Theorem 5.14 does not give an estimate of the size of the set system on $\{1, \ldots, n\}$ to have a non zero discrepancy. Roth [48] by his $\frac{1}{4}$-theorem gives us

a lower bound $\Omega(n^{\frac{1}{4}})$ on discrepancy on arithmetic progressions on $\{1, 2, ..., n\}$ for total 2-coloring. The size of the set system is $\theta(\sqrt{n})$. $D_{\mathcal{C}_T}(S) \neq 0 \implies D_{\mathcal{C}}(S) \neq 0$ but the converse is not true. So, there is no known lower bound on the number of hyper-edges in a hypergraph for having non-zero discrepancy for partial 2-colorings.

For k-uniform hypergraphs it has been shown in [2] that $\Omega(\frac{\log\log k}{\log\log\log k})$ sized set system is enough for having non zero discrepancy with total 2-coloring.

5.5.2 Relation with Modular Composition

Let n be a positive integer which expresses the length of the each equal length strings u and v. We denote the set of first n positive integers by $[n]$. A finite arithmetic progression on $[n]$ is denoted as $AP_n(a, d)$ where the arithmetic progression looks like $\{a, a + d, a + 2d, \ldots, a + \lfloor\frac{n-a}{d}\rfloor d\}$.

Theorem 5.15. The set system $APC(k, n) = \{AP_n(a, d) \mid 1 \leq a < d \leq m\}$ forms a k-regular hypergraph on $[n]$.

Proof. Let any number $i \in [n]$ can be written as $i = x + sd = y + rd$ for some $1 \leq x, y \leq d \leq k$ and $x, y, s, r \in \mathbb{N}$. Then let $x > y$ which means $x - y = d(r - s) > 0$. But that is not possible since $d \geq x, y \implies d > x - y \implies d(r - s) > x - y$. Hence i can be uniquely written as $x + sd$ when x and s are variables and $x \leq d$. In $APC(k, n)$ there are exactly k options to choose from for every entry in $[n]$. So, every entry in $[n]$ occurs at exactly k sets in $APC(k, n)$. Hence $APC(k, n)$ forms a k-regular hypergraph on $[n]$. □

Robson used permutation machine for separating binary words. The idea was to look for odd number of '1's at certain subsequence of two words. The elements of the subsequence formed arithmetic progressions. An extension of that idea is to look for total number of '1's in certain arithmetic progressions related to these strings.

Here we shall show that the problem of finding non zero discrepancy with respect to partial colorings to a hypergraph of the set system $APC(k, n)$ can be reduced to an instance of finding distinct k-modular compositions for every pair of n length strings and vice versa.

If we consider two n length binary words and write one above another then we can think of them as a collection of n 2×1 matrices. Each of them would be from the column vectors of the following matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

If we treat $[0\ 1]^t$ as -1, $[1\ 0]^t$ as 1, both $[0\ 0]^t$ and $[1\ 1]^t$ as 0 then for a certain set of arithmetic progression A in two words, the number of '1's are same in every arithmetic progression means we have a perfect partial coloring χ on A or $\mathcal{D}_\chi(A) = 0$.

Now the question is - what is the lower bound and upper bound on m such that $D_{\mathcal{C}}(APC(m,n)) > 0$?

We have have one mathematical confirmation that if

$$\prod_{i=1}^{m}(1+\frac{n}{i})^i > 2^{[n]}$$

then existence of a perfect partial coloring is not guaranteed for $APC(m,n)$.

Lemma 5.16. Let $APC(m,n)$ be a set system on $[n]$ then there exists $m \in O(\frac{\sqrt{n}}{\log n})$ such that $D_{\mathcal{C}}(APC(m,n)) > 0$.

Proof. Robson [47] proved that there exists an arithmetic progression $A \in APC(m,n)$ where $m \in O(\frac{\sqrt{n}}{\log n})$ such that number of '1's in two arbitrary same length binary strings will be different at the positions given by A. In fact he proved more specifically that number of '1's in one string will be odd and in another will be even. This proves the discrepancy $D_{\mathcal{C}}(APC(m,n)) > 0$. □

Lemma 5.17. Let $APC(m,n)$ be a set system on $[n]$ then $D_{\mathcal{C}}(APC(m,n)) > 0$ if $m \in \Omega\left(\sqrt{\frac{n}{\log n}}\right)$.

Proof. In the first part of the proof of the Theorem 5.5 we have seen that to for any pair of strings $u, v \in \{0,1\}^n$ with $\text{mod}_1^l(u) \neq \text{mod}_1^l(v)$, l has to be in $\Omega\left(\sqrt{\frac{n}{\log n}}\right)$. It

also means for any partial coloring $\chi \in \mathcal{C}$ $D_{\mathcal{C}}(APC(m,n)) > 0$ if $m \in \Omega\left(\sqrt{\frac{n}{\log n}}\right)$. □

5.5.3 Generalization to Arbitrary Alphabet

Although the idea of discrepancy in arithmetic progression can be used to separate binary strings the idea can be used to separate words from any alphabet.

Let Σ be any alphabet and u, v are two strings with the same length. Now for each letter $c \in \Sigma$ and again consider a $2 \times n$ matrix h where $h_c^{(u,v)}(1,i) = 1$ iff $u(i) =$'c' and $v(i) \neq$ 'c', $h_c^{(u,v)}(2,i) = 1$ iff $u(i) \neq$'c' and $v(i) =$'c', $h_c^{(u,v)}(1,i) = 0, h_c^{(u,v)}(2,i) = 0$ for all other cases. Once we have the matrix $h_c^{(u,v)}$ then we can proceed as before and finding perfect partial coloring on the sequence $H_c^{(u,v)}$ where $H_c^{(u,v)}(i) = 1$ iff $h_c^{(u,v)}(1,i) = 1$ and $h_c^{(u,v)}(2,i) = 0$, $H_c^{(u,v)}(i) = -1$ iff $h_c^{(u,v)}(1,i) = 0$ and $h_c^{(u,v)}(2,i) = 1$, $H_c^{(u,v)}(i) = 0$ for all other cases. Finding a perfect partial coloring in arithmetic progressions in $H_c^{(u,v)}$ will ensure that there are same numbers of 'c' in u and v at indices given by those arithmetic progressions.

5.5.4 Remark

In this chapter we have used the existing result of the separating words problem to find the condition on the set systems to have non zero discrepancy with respect to partial colorings. Any arbitrary problem of finding non zero discrepancy with respect to partial colorings will be helpful to formulate instance of separating word problem if the set system can be made in such a way that every set from the set system contains the elements from the set of points in a "regular" way. By "regular" we mean that the set builder rule for constructing the set system must be in a form such that a DFA can capture it. Any further improvement of such the lower bound on Lemma 5.17 will automatically result in an improvement in the separating word problem.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Chapter 6

Summary and Related Open Problems

We have investigated factors, more specifically multiset of factors upto length k, that is $m_{[1,k]}(u)$ of a string u.

A string u can be reconstructed from $m_{[1,k]}(u)$ ambiguously.. If the reconstructed string is not u then that reconstructed string is called k-abelian equivalent to u or an ambiguous reconstruction of u from $m_{[1,k]}(u)$.

In Chapter 2 we have proved that it is enough to check k and $(k-1)$-length factor multisets of two string to identify if they are k-abelian equivalent or not.

In Chapter 3, it is shown that the reconstruction of u from $m_{[1,k]}(u)$ is similar to the process of reconstructing u from the multiset of k-length factors of u. We have also shown the relation between the number of different ambiguous reconstructions using these two methods. We give the lower bound of maximum number of ambiguous reconstructions of any string for $k = 2$. For the special value of $k = \lfloor \frac{|u|}{2} \rfloor$, the maximum number of ambiguous reconstructions of any string is proved to be three. In this context we can have a look at the following open problem:

Open Problem 6.1. Given an alphabet Σ where $|\Sigma| = \alpha$ and a string length n with a factor size $k < n$ find $\max_{u \in \Sigma^n} |\mathbb{M}_k(u)|$ in terms of n, k and $|\Sigma|$.

Although BEST theorem and subsequently Theorem 3.2 gives us a way to enumerate the reconstructions from $m_{[1,k]}(u)$ it does not give the asymptotic formula

in terms of n and k. If we consider a particular kind of graph then we may look for an asymptotic formula of the enumeration having some properties of the graph as parameters. For example McKay and Robinson in [35] found the asymptotic number of eulerian cycles in the complete graph. Isaev found in [26] asymptotic number of Eulerian circuits in complete bipartite graphs. The asymptotic number of eulerian cycles in a de Brujin graph G_k on binary alphabet is known [9] to be $2^{k-1} - k$. In [36] we get efficient randomized schemes to sample and approximately count Eulerian orientations of any Eulerian graph. Also in [37] there is mention of maximizing the number of aorborescences for graphs. The author has kept it as an open problem for general multidigraph. This direction may give us the idea of what kind of strings have maximum number of reconstructions.

In Chapter 4, a generalisation on k abelian equivalence of words has been introduced. We can look $m_{[1,k]}$ as different factors upto length k of the string u and the number of ways each them can be in the string u as factors. By k-factor precedence data of a string u we mean different ordered pairs of k-length factors of the string u and for each of them the number of different ways the first component occur as a factor before the second component as a factor in the string u. Different properties of k-factor precedence data were investigated. The question of reconstruction of a word from its k-factor precedence data has however remained a topic for further investigation. We present an open problem in this context:

Open Problem 6.2. One can represent the 1-factor precedence data or precedence data of a string s by the matrix OP_1^s together with the ordered set F_1^s of different letters forming s. So, is there any polynomial time algorithm that can identify whether for any given $n \times n$ matrix M, with $n > 2$ and an ordered set F of n letters, there is a string v such that $OP_1^v = M$ and $F_1^v = F$?

We have tried to characterize the special type of permutations on a string u, that yields k-factor precedence equivalent strings. For a special subclass of such permutations it is proved that two different strings needed to be of length at least $3k + 1$ such that one can apply that kind of permutation to obtain one string from the other. On this note we present two open problems:

Open Problem 6.3. If $OP_k(u) = OP_k(v)$ then is it guaranteed that v can be obtained from u by applying one or several k-mirror transpositions?

Open Problem 6.4. Is there any example of a pair of different strings u and v with $|u| = |v| < 3k + 1$ if $OP_k(u) = OP_k(v)$?

In Chapter 5, we have modified another generalization on k-abelian equivalence of words. l-modular factor composition of a string u is the union of multisets of factors of length up to l, which occur at u as factors at positions with congruent to a mod d with $0 \leq a < d \leq l$. We have investigated in different properties of l-modular factor composition of a word and its relation with k-abelian equivalence of words. We extended this idea to get yet another generalisation named l-modular factor precedence composition of words. This composition of a given string u is the set of ordered pairs of l length factors of u occurring as factors of at positions congruent to a mod d with $0 \leq a < d \leq l$ and for each of them the number of different ways the first component occur as a factor before the second component as a factor in the string u. We have found an upper bound on the least length needed for a pair of strings to have the same l-modular factor precedence data. In this context we present another open problem:

Open Problem 6.5. Is there any improvement on $l = O(\sqrt{n})$ such that $\text{mod}_l(u) \neq \text{mod}_l(v)$ or $\text{premod}_l(u) \neq \text{premod}_l(v)$ for any pair of strings u and v of length n?

This problem is directly related to the famous open problem in the context of separating words problem. We also mention that problem:

Open Problem 6.6. Is there any better upper bound than $O(n^{\frac{2}{5}} \log^{\frac{3}{5}}(n))$ in the least number of states in a DFA separating any pair of strings of length n?

We already know from Chapter 2 that if $m_{[1,k]}(u) = m_{[1,k]}(v)$ then u and v are said to be in the same class generated by equivalence relation $\overset{k}{\underset{1}{\equiv}}$ on set of all strings.

In a similar manner, let us define the following relations:

- $\left(\overset{prec}{\underset{k}{\equiv}}\right)$ We say for two strings u and v, $u \overset{prec}{\underset{k}{\equiv}} v$ if and only if $OP_k(u) = OP_k(v)$.

- $\left(\overset{mod}{\underset{k}{\equiv}}\right)$ We say for two strings u and v, $u \overset{mod}{\underset{k}{\equiv}} v$ if and only if $\text{mod}_k(u) = \text{mod}_k(v)$.
- $\left(\overset{premod}{\underset{k}{\equiv}}\right)$ We say for two strings u and v, $u \overset{premod}{\underset{k}{\equiv}} v$ if and only if $\text{premod}_k(u) = \text{premod}_k(v)$.

From Chapter 4 we have $u \overset{mTr}{\underset{k}{\equiv}} v$ if and only if u can be obtained from v by applying one or several k-mirror transpositions.

In general let two strings are related by R_k where the nature of the relation depends on the value of k. Let $\mathfrak{s}(u, v)$ = minimum value of k such that $u\tilde{R}_k v$. By $u\tilde{R}_k v$ we mean u and v does not belong to the same class formed by the relation R_k on set of all strings. Also $S(n) = \max_{|u|\leq n, |v|\leq n} \mathfrak{s}(u, v)$. In Table 6.1 we summarize different compositions of strings based on its factors.

Relation R_k	Lower bound on $S(n)$	Upper bound on $S(n)$
$\overset{k}{\underset{1}{\equiv}}$	$\lfloor \frac{n}{2} \rfloor + 1$	$\lfloor \frac{n}{2} \rfloor + 1$
$\overset{mTr}{\underset{k}{\equiv}}$	$\lfloor \frac{n-1}{3} \rfloor + 1$*	$\lfloor \frac{n-1}{3} \rfloor + 1$*
$\overset{prec}{\underset{k}{\equiv}}$	$\lfloor \frac{n-1}{3} \rfloor + 1$*	no precise bound known ‡
$\overset{mod}{\underset{k}{\equiv}}$	$\Omega\left(\sqrt[3]{\frac{n}{\log n}}\right)$*	no precise bound known †
$\overset{premod}{\underset{k}{\equiv}}$	$\Omega\left(\sqrt[4]{\frac{n}{\log n}}\right)$*	no precise bound known †

*Results proved in this thesis.
‡Best upper bound is $\lfloor \frac{n}{2} \rfloor + 1$.
†Best upper bound is Robson's bound $O(\sqrt{n})$.

Table 6.1: Comparison of different compositions of strings based on their factors

References

[1] C. Acquisti, G. Poste, D. Curtiss, and S. Kumar. Nullomers: Really a Matter of Natural Selection? *PLoS ONE*, 2(10), 2007.

[2] N. Alon, D. J. Kleitman, C. Pomerance, M. Saks, and P. Seymour. The smallest n-uniform hypergraph with positive discrepancy. *Combinatorica 7 (2) (1987) 151–160*, 7(2):151–160, 1987.

[3] Jacek Baewicz and Marta Kasprzak. Complexity of DNA sequencing by hybridization. *Theoretical Computer Science*, 290(3):1459 – 1473, 2003.

[4] W. Bains and G. Smith. A novel method for nucleic acid sequence determination. *Journal of Theoritical Biology*, 135:303–307, 1988.

[5] J. Berstel, A. Lauve, C. Reutenauer, and F. V. Saliola. *Combinatorics on words ; Christoffel words and repetitions in words.* American Journal of Mathematics, 2009.

[6] F. Blanchet-Sadri. *Algorithmic Combinatorics on Partial Words.* Chapman & Hall/CRC, 2008.

[7] Alexander B. Chetverin and Fred Russell Kramer. Sequencing of pools of nucleic acids on oligonucleotide arrays. *Biosystems*, 30(1-3):215–231, 1993.

[8] James Currie, Holger Petersen, John Michael Robson, and Jeffrey Shallit. Separating words with small grammars. *Journal of Automata, Languages and Combinatorics*, 4:101–110, 1999.

[9] N. G. de Bruijn. A Combinatorial Problem. *Koninklijke Nederlandse Akademie van. Wetenschappen*, 49, 1946.

[10] Erik Demaine, Sarah Eisenstat, Jeffrey Shallit, and David Wilson. Remarks on separating words. In Markus Holzer, Martin Kutrib, and Giovanni Pighizzini, editors, *Descriptional Complexity of Formal Systems*, volume 6808 of *Lecture Notes in Computer Science*, pages 147–157. Springer Berlin, 2011.

[11] R. Drmanac, I. Labat, I. Brukner, and R. Crkvenjakov. Sequencing of megabase plus dna by hybridization: Theory of method. *Genomics*, 4:114–128, 1989.

[12] van Aardenne Ehrenfest and N. G. de Bruijn. Circuits and trees in oriented linear graphs. *Simon Stevin: Wis- en Natuurkundig Tijdschrift*, 28:203217, 1951.

[13] Neven Elezovic. Asymptotic Expansions of Central Binomial Coefficients and Catalan Numbers. *Journal of Integer Sequences*, 17(14.2.1), 2014.

[14] H. Fleischner. Eulerian Graphs and Related Topics: Chapter IX- Eulerian Trails How Many? *Annals of Discrete Mathematics*, 50:X.1–X.110, 1991.

[15] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[16] I. J. Good. Normal Recurring Decimals. *Journal of the London Mathematical Society*, 21(3), 1946.

[17] P. Goralčík and V. Koubek. On discerning words by automata. In *Automata, Languages and Programming*, volume 226 of *Lecture Notes in Computer Science*, pages 116–122, 1986.

[18] D. Gusfield. *Algorithms on Strings, Trees, And Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[19] Greg Hampikian and Tim Andersen. Absent sequences: nullomers and primes. In *Pacific Symposium on Biocomputing*, volume 12, pages 355–66, 2007.

[20] S. Hannenhali, P.A. Pevzner, H. Lewis, S. Skiena, and W. Feldman. Positional sequencing by hybridization. *Computer Applications in Biosciences*, 12:19–24, 1996.

[21] Julia Herold, Stefan Kurtz, and Robert Giegerich. Efficient computation of absent words in genomic sequences. *BMC Bioinformatics*, 9(167), 2008.

[22] Carl Hierholzer. Ueber die Moglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. *Mathematische Annalen*, 1873.

[23] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[24] Joan P. Hutchinson and Herbert S. Wilf. On Eulerian Circuits and Words with Prescribed Adjacency Patterns. *Journal of Combinatorial Theory*, 18(A):80–87, 1975.

[25] Lucian Ilie and William F. Smyth. Minimum unique substrings and maximum repeats. *Fundamenta Informaticae*, 110(1-4):183–195, Jan 2011.

[26] M.I. Isaev. Asymptotic number of Eulerian circuits in complete bipartite graphs. In *Proc. 52-nd MFTI Conference (in Russian)*, page 111114, 2009.

[27] J. Howard Johnson. Rational equivalence relations. *Theoretical Computer Science*, 47:39–60, November 1986.

[28] Dieter Jungnickel. *Graphs, Networks and Algorithms*, volume 5 of *Algorithms and Computation in Mathematics*. Springer, Fourth edition, 2013.

[29] Juhani Karhumaki, Aleksi Saarela, and Luca Q. Zamboni. On a generalization of Abelian equivalence and complexity of infinite words. *Journal of Combinatorial Theory, Series A*, 120:2189–2206, September 2013.

[30] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer, 2008.

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

[32] M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, 2002.

[33] M. Lothaire. *Applied Combinatorics on Words*. Cambridge University Press, 2005.

[34] Y. Lysov, V. Florent'ev, A. Khorlin, K. Kharapko, V. Shik, and A. Mirzabekov. DNA Sequencing by Hybridization with Oligonucleotides. *Doklady Academy Nauk USSR*, 303:1508–1511, 1988.

[35] Brendan McKay and Robert W. Robinson. Asymptotic enumeration of eulerian circuits in the complete graph. *Combinatorica*, 10(4):367377, 1995.

[36] M. Mihail and P. Winkler. On the number of Eulerian orientations of a graph. *Algorithmica*, 16:402–414, 1996.

[37] Wendy Myrvold and Kathryn L. B. Wood. Counting Spanning Out-trees in Multidigraphs. University of Victoria, June 2000.

[38] P. A. Pevzner. l-Tuple DNA Sequencing: Computer Analysis. *Journal of Biomolecular Structural Dynamics*, 7(1):63–73, Aug 1989.

[39] P. A. Pevzner. DNA Physical Mapping and Alternating Eulerian Cycles in Colored Graphs. *Algorithmica*, 13:77–105, 1995.

[40] Claribet Piña and Carlos Uzcátegui. Reconstruction of a word from a multiset of its factors. *Theoretical Computer Science*, 400(1-3):70–83, 2008.

[41] Jean-Eric Pin. On reversible automata. In Imre Simon, editor, *LATIN '92*, volume 583 of *Lecture Notes in Computer Science*, pages 401–416. Springer Berlin Heidelberg, 1992.

[42] G. Rauzy. Suites a termes dans un alphabet fini, Seminar on number theory, 1982-1983. volume 25, University of Bordeaux I, Talence, 1983.

[43] O. Razgulyaev, A. Rubinov, M. Gelfand, and A. Chetverin. Sequencing potential of nested strand hybridization. *Journal of Computational Biology*, 2(2):383–395, 1995 Summer.

[44] Michel Rigo and Pavel Salimov. Another generalization of abelian equivalence: Binomial complexity of infinite words. In Juhani Karhumki, Arto Lepist, and Luca Zamboni, editors, *Combinatorics on Words*, volume 8079 of *Lecture Notes in Computer Science*, pages 217–228. Springer Berlin Heidelberg, 2013.

[45] Fred S. Roberts. *Graph Theory and Its Applications to Problems of Society*, volume 29 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Rutgers University, 1978.

[46] J. M. Robson. Separating strings with small automata. *Information Processing Letters*, 30(4):209–214, 1989.

[47] John Michael Robson. Separating words with machines and groups. *Informatique Théorique et Applications*, 30(1):81–86, 1996.

[48] K. F. Roth. Remarks concernig integer sequences. *Acta Arithmetica*, 9:257–260, 1964.

[49] A.R. Rubinov and M.S. Gelfand. Reconstruction of a string from substring precedence data. *Journal of Computational Biology*, 2(2):371–381, 1995.

[50] Flye Sainte-Marie. Solution to question nr. 48. *L'intermdiaire des Mathmaticiens* , 1:107–110, 1894.

[51] J. Shallit. *A Second Course in Formal Languages and Automata Theory*. Cambridge, 2008.

[52] Jeffrey Shallit and Ming-Wei Wang. Automatic complexity of strings. *Journal of Automata, Languages and Combinatorics*, 6(4):537–554, April 2001.

[53] E. Southern. GB8810400. Technical report, United Kingdom patent application, 1988.

[54] Joel Spencer. *Ten Lectures on the Probabilistic Method*, volume 64 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 2 edition, 1994.

[55] James A. Storer. *Data Compression: Methods and Theory* , volume 13 of *Principles of Computer Science*. Computer Science Press Inc.(a subsidiary of W. H. Freeman Press), New York, NY, USA, 1988.

[56] E. Ukkonen. Approximate String Matching With q-grams and Maximal Matches. *Theoretical Computer Science*, 92:191–211, 1992.

[57] M.N. Vyalyi and R.A. Gimadeev. Separating Words by Occurrences of Subwords. *Journal of Applied and Industrial Mathematics*, 8(2):293–299, 2014.

[58] Zong-Da Wu, Tao Jiang, and Wu-Jie Su. Efficient computation of shortest absent words in a genomic sequence. *Information Processing Letters*, 110(14-15):596–601, 2010.