Informatique

UNIVERSITY OF STRASBOURG

Mannheim University of Applied Sciences

DOCTORAL SCHOOL MATHEMATICS, INFORMATION SCIENCES AND ENGINEERING

CSTB, UMR-7357

THESIS presented by :

Martin STARMAN

defended on: **11th December 2020**

to obtain the grade of: **Doctor of the University of Strasbourg**

Discipline/ Speciality: Computer Science

Circular codes in the evolution of the genetic code

THESIS supervised by :
M. MICHEL Christian Professor, University of Strasbourg

THESIS co-director :
STRÜNGMANN Lutz Professor, Mannheim University of Applied Sciences

REFEREES :
GIANNERINI Simone Professor, University of Bologna
DENISE Alain Professor, University Paris-Saclay

OTHER MEMBERS OF THE JURY :
FIMMEL Elena Professor, Mannheim University of Applied Sciences
SERENI Jean-Sébastien Professor, University of Strasbourg

Contents

Acknowledgment

I would like to thank the following people, without whom I would not have been able to complete this work:

First I would like to thank Chrisian Michel and Lutz Strüngmnn for supervising my dissertation. Both have inspired my work and provided guidance and feedback throughout this project. An equally great encouragement came from Elena Fimmel, who was a great support for my research. Markus Gumbel, who advised me on the analysis of the gene sequence also deserves my gratitude. I would also like to thank the whole COMMBIO team for their support. Without them the work would not have been successful. All the co-authors of the published papers I contributed to have supported me in my development and deserve to be mentioned in my acknowledgment. Additionally, I would like to thank the whole defense jury Alain Denise, Ellena Fimmel, Simone Giannerini, Chrisian Michel, Jean-Sébastien Sereni and Lutz Strüngmnn for their effort and support. Many thanks to all the proofreaders, Carolin Zorell and Sarah Willenbücher, whom I demanded a lot from. Thanks also to my parents who have always motivated and supported me. And finally, I would like to thank my beloved wife Julia, who always supported me despite my many hours in the home office. Only thanks to her tireless help as a proofreader and loving support I was able to finish this work.

Thesis abstract

Circular codes in the evolution of the genetic code

The problem that this work addresses is how to retrieve, maintain and synchronize the correct reading frame during the translation process. Translation is the process by which the ribosome decodes the messenger RNA (mRNA a sequence of nucleotides $\{A, C, G, T\}$) as codons (words of 3 nucleotides) to create a specific amino acid chain that later folds into a protein. Unfortunately, the mRNA can be decoded in three reading frames 0, +1 and +2. Yet, only frame 0 as correct reading frame encodes the Information for the synthesis of proteins. Usually, the correct reading frame is indicated by a start signal. First practical evidence of a genetic model which is able to retrieve the correct reading frame is the so-called X-code. The X-code is a set of 20 codons that was discovered by a statistical analysis of genes of different species [1]. Concatenations of words of this code appear preferentially in genes in frame 0. Such a concatenation is called a motif. The proportion of the motifs found in genes compared to non-coding regions was significantly high. Astonishingly, it turned out that the X-code is a circular code. A circular code is a block code and defined so that a concatenation of words of a code written on a cycle can only be decomposed in words of the code in one reading frame. The advantages of circular genetic codes are incomparable: firstly, even without any kind of start signal the correct reading frame can be ensured. Secondly, the reading frame is automatically retrieved within a window of only a few nucleotides [24].

The X-code also has strong mathematical properties. In detail, X is maximal as it cannot be contained in a circular code of higher cardinality, self-complementary since for any codon in X the responding anti codon is also in X and C^3, i.e. all circular permutations of X are also maximal circular codes. In total there are 216 different maximal, self-complementary and C^3 codes. These 216 codes can be divided into 27 equivalence classes. These classes of codes have been studied by representing them by directed graphs and then applying deep graph theory [20]. Block codes which are closely related to the circular codes are the comma-free codes. These codes are a more restrictive variant of circular codes in which the

reading frame is found immediately after only one word. Both kinds of codes can be represented in a directed graph. Such a graph has a directed arc between all prefixes and concatenated suffixes of all words of the represented code. The graph is defined so that if it is acyclic, the represented code must be circular. Moreover, if the graph is acyclic and its longest path has a length of two at the most, the code is even comma-free [27].

Circular codes are also supposed to play a role in an evolutionary context. Most theories of evolution state that the modern genetic code has several ancestors. It has been proposed that circular and comma-free codes could be a selection factor that guides the search for these hypothetical ancestor code [59], [1]. Some of these theories describe an evolutionary ancestor to the modern genetic code that consisted not only of trinucleotides, but also of dinucleotides, tetranucleotides or combinations of them (see [39, 3, 68, 63, 75, 78]). Even a hypothesized ancestor code with a different alphabet has been proposed. This initiated the study of circular codes in a more general context with words of arbitrary finite length over general alphabets [28, 22] and relates the theory of circular codes to signal processing and general information theory.

Contribution

The statistical evidence shows the existence of a circular code-related model in RNA and DNA sequences. Thus, the foremost objective of this thesis is the expansion of the knowledge of circular codes. The first part of the thesis investigates fundamental properties of the circular code family from a mathematical biology perspective. In the second part a software package is developed, and the properties are used in algorithms to refine the identification of such codes in the genetic code.

Properties of circular codes

The first chapter of the thesis contains two sections. The first section is referring to the published article [24] and uses the codes which are elaborated by extracting data from genes. Most results in this section describe the properties of the 216 maximal self-complementary C^3 codes. The second section uses a more general focus on circular codes. The issues in focus of this chapter is the minimum number of nucleotides to ensure the correct reading frame, the properties of self-complementary codes and the maximum size of circular codes of different word length ℓ and alphabet. One of the most important findings is the reading frame number. This number indicates the minimum length (number of nucleotides) of a code motif to ensure the correct reading frame. In addition, we have succeeded in deriving the reading frame number from the longest path in the associated graph. Hence, it can be used as an effortless calculable quality feature of codes. In more detail we

characterize all possible patterns of longest paths in self-complementary circular codes. Furthermore, we were able to fully classify all graphs associated with a self-complementary code of size at least 18 using recognizable conditions. We also discovered a combinatorical construction of not self-complementary codes of size less than 18 where the associated graph satisfies the conditions of self-complementary codes. This served as a counterexample for the conjecture that these conditions in a graph force the represented code of any size to be self-complementary. The result that a code with a size of at least 18 must be self-complementary if the associated graph has the recognisable conditions is new and has not yet been published. The second section of this chapter is also unpublished. This section starts with an improved and easily scriptable algorithm to calculate the maximum size of circular codes for all word lengths and alphabets. Next, it illustrates a mathematical model to transform circular codes into circular codes of different word length and/or alphabet. A similar model could have had an influence on the evolution of the hypothetical ancestor code. Finally, it also depicts a method to separate the 60 codons (excluding the four identity codons AAA, CCC, ...) in four so called comma-free subsets. These can be used to support the proposal in [59].

Circular Tessera codes

The second chapter presents circular Tessera codes. Most results were published in article [28]. The Tesserae are a subset of tetranucleotides. These specified tetranucleotides have been developed to situate the symmetry as a code property in the evolution of the genetic code. The model presented by Gonzalez, Giannerini and Rosa [40] supports the theory that the Tessera code is one possible step in the evolutionary process of the genetic code. Therefore, a combination of the Tessera code theory and the circular code theory could be an evidence for each theory and explain their role in RNA sequences.

In this thesis it is shown that circular Tessera codes can be divided into four equivalence classes. The equivalence classes are orbits using a group of four bijective symmetry transformations. Each class is represented by a disjoint component of the representing graph. Based on this discovery, an algorithm is presented that constructs all possible circular Tessera codes. With this algorithm several properties of circular Tessera codes can be identified. The most important properties that are obtained are (A) that the graph associated with a self-complementary code is fully characterized by the use of recognizable conditions and (B) that the longest path in an associated graph is either 1, 2 or 3. In this chapter we also show a refined version of the algorithm, which allows to construct all codes exactly once for every possible code length. This unpublished algorithm describes a full construction of all circular Tessera codes using group theory and advanced methods of graph theory and combinatorics.

The relation between k-circularity and circularity of codes

The third chapter of the thesis refers to the published article [22]. There we introduce the k-circular codes, a new class of block codes. k-circular codes belong to the circular code family. Like the circular codes, such a code is a frame shift error detecting code and consists exclusively of words of block length ℓ. It is a weakened version of the circular code. If a circular code can find the correct reading frame in every word written on a circle, a k-circular code requires a concatenation of a maximum of k words to reliably ensure the reading frame. Therefore, a concatenation of $k+1$ words written on a circle could be read in more than one reading frame. Thus, a k-circular code does not have to be $k+1$-circular but must be $k-1$-circular. The class of k-circular codes contains both circular and comma-free codes as subclasses for every given k.

To say whether a code is circular, one needs to check if any concatenation of infinite words of a code written on a cycle can only be decomposed in words of the code in one reading frame. Consequently, a code is circular if, and only if it is a ∞-circular code. This makes it an undecidable problem. To make this problem decidable, we were able to identify a number $0 < k(n,\ell) < \infty$, so that a code is only circular if it is $k(n,\ell)$-circular, where n is the cardinal number of the alphabet and ℓ is the word length of the code. We also show that the upper bound $k(n,\ell)$ is sharp. Hence, we demonstrate a construction algorithm for a $k(n,\ell)-1$-circular code which is not circular for all $\ell, n \in \mathbb{N}$. This proves the sharpness of $k(n,\ell)$. In conclusion, this tremendously reduces the combinatorial complexity to check if a code is circular.

Algorithms and tools to identify codes in Sequences

The final chapter of this thesis presents practical applications of the theoretical results obtained in the previous chapters and is entirely new. It is separated into two main sections. The first section presents a software package GCATR, which is an implementation of all important tools and methods that support the investigation of theories in genetic data. The aim of the second section is to introduce new methods for detecting error-detecting codes in DNA sequences.

The software GCATR (Genetic Code Analysis Toolkit R) is an R package which includes all important functions to work with circular codes and their relatives. The core is written in C++, which makes it highly performant. The R wrapper further allows an easy usage. Even the development of parallel algorithms can be achieved easily with the R native commands. The thesis gives a short overview of the used architecture and the developed algorithms in GCATR. Finally, it summarises the functionalities of the package.

In the second section, two methods to obtain new evidence of circular codes

in genetic coding are introduced. The methods are theoretical models of a possible reading frame detection in the ribosome using a new approach different from circular code theory. In order to optimise the retrieved code with respect to the methods, a hillclimber algorithm is used. As data, a randomly generated sample set of coding sequences (CDS) of different species is used. Surprisingly, the results of the hillclimber indicate that the returned codes are always either k-circular or even circular codes. Moreover, the codes obtained have a coverage above average in the tested sequences - a hint to a possible role that they might play in the protein coding process.

The thesis finally closes with a discussion of the reliability of the evidence of circular codes in the evolutionary process of the genetic code.

Chapter 1

Introduction

> " *We understand biological phenomena only when we have invented machines with similar properties*"
> → Maynard Smith, 1986

Almost 35 years later, Maynard Smith's words continue to be an inspiration for researchers worldwide. In the same spirit, this thesis focuses on the construction of parts of a machine to promote our understanding of the evolution of the genetic information system. It is the general opinion among researchers working on the evolution of the genetic code that such an efficient system cannot have appeared spontaneously. It can be approximated that a cell contains 978 *million base pairs* per *picograms*, *i.e.* 0.489 *gigabyte* of heritage information [16]. Such a tight system must be perfectly organized. This leads to the assumption that an error-correcting code evolved as a subcode inside in the genetic code during the course of evolution. This new potential guidance factor of the evolution has already led to landmark studies. Biomathematicians as well as computer scientists have advanced the knowledge about such error-correcting codes and their appearance in the genetic code. In this dissertation, we follow theories of error-correcting codes in the evolution and find mathematical answer to important questions regarding such codes, which had remained open so far. Before presenting the developed components of a mathematical model that reconstructs the development of a hypothetical block code based system, the introductory chapter provides an overview of the biological motivation and mathematical tools used in the research on which this thesis is based on.

This chapter is separated into two sections. The first Section 1.1 summarizes the biological foundation of this thesis, *i.e.* the structure of cells, the process of protein synthesis and the most plausible evolutionary theories currently considered by scientists. The fundamental basics can be found in the book [49]. The second Section 1.2 introduces codes as mathematical construct. After a general definition

of codes, circular codes and their relatives are specified. In addition, methods from group theory and graph theory are presented that are used to investigate circular codes from a mathematical point of view. The chapter closes with the presentation of the X-code, *i.e.* the link between code theory and genetic information processing.

1.1 Biological motivation

The complexity of the mechanism behind the inheritance of genetic information is an evolutionary masterpiece, which is continuously progressing. It is impossible to determine the stage of development we are currently at. Even the process of evolution up to the present day, the question of the origin of life is controversially discussed. There is, however, consensus on the time life on earth began: About 4.2 billion years ago [4] the first cells were the origin of all life. Based on today's idea of evolution, it must be assumed that the cells have been vastly advancing ever since. Thus, the first cells, the common ancestor of all life, must have had very primitive and simple genetic information storage, whereas today's cells have the ability to store a tremendous amount of information. Presently, there are three known forms of life: archaea, bacteria and eukaryotes. They are grouped into two different types of cells: prokaryotic and eukaryotic cells. The prokaryotic cell occurs in all living organisms under the domains archaeae and bacteria, while all organisms grouped as eukaryotes, including human beings, have eukaryotic cells. One representative of each cell type is illustrated in Figure 1.1. While the prokaryotic cells have a cyclic DNA the eukaryotic cells have a non-cyclic DNA organized in chromosomes. The chromosomes are separated from the rest of the cell in the nucleus. In comparison, in prokaryotic cells, DNA is not separated from the rest of the cell. Another difference between these two types of cells is that only in the eukaryotic cell the mRNA is preprocessed before being sent to the ribosome. Even though prokaryotic and eukaryotic cells differ in various aspects we will focus on the similarities.

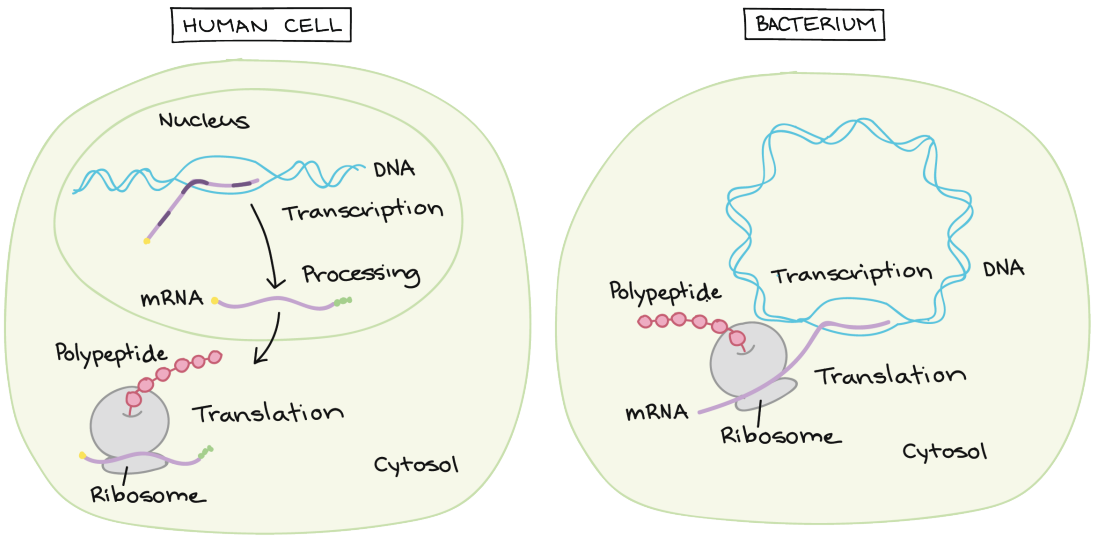


Figure 1.1: A schematic representation of the process from transcription to translation. The human cell, which belongs to the eukaryotic cell type on the left side, is compared with a bacterial cell, a representative of the prokaryotic cell type, on the right side. (Picture taken from https://www.khanacademy.org/science/high-school-biology/hs-molecular-genetics/hs-rna-and-protein-synthesis/a/intro-to-gene-expression-central-dogma)

Both cell types have a DNA double helix. This basic element of genetic information storage is made of two strands winding around each other. Another common feature is the messenger RNA (mRNA), which is the result of the process called transcription. This mRNA transports the genetic information, gathered from the DNA during the transcription, to the ribosome. The ribosome then runs the translation. Translation is the process during which the ribosome decodes the mRNA to create a specific amino acid or polypeptide chain that later folds into a protein. The pre-translation mRNA processing ignored, eukaryotic and prokaryotic cells share the main aspects of the information processing system. The complexity of such a system is impressive, especially acknowledging the fact that it was developed in the process of evolution.Here it must be assumed that we have not yet reached the evolutionary peak. In order to assess the potential future cell structure, understanding past developments is key. Fortunately, we may be in the position to reconstruct the past 4.2 billion years of evolution of the living cell. This is due to the fact that there are two different types of cells in three different domains of life to compare with each other, and we know that they must have a common ancestor. Before going deeper into the foundations of the evolution of life, it is necessary to understand the current knowledge of molecular biology

about genetic information. [1]

1.1.1 RNA and DNA

The "double helix" has become known all over the world ever since James Watson published his famous book "The Double Helix: A Personal Account of the Discovery of the Structure of DNA" in the year 1968. The double helix consists of two polynucleotide chains that coil around each other. These chains contain monomeric molecules (monomeric molecules can chain with other monomeric molecules). These molecules are called nucleotides and are built from a sugar, a phosphate group and, as an information-coding-unit, one of four nucleobases: *adenine* (A), *guanine* (G), *cytosine* (C) or *thymine* (T). The two strands in the double helix are connected by hydrogen bonds between so-called complementary nucleobases. Where A is complementary to T and C is the complement of G. These same hydrogen bonds are also used during the process of transcription, when the messenger RNA copies and transforms the information of the DNA. While DNA consists of two strands, RNA is a single stranded structure of monomeric molecules. Similar to DNA, the molecules in RNA are called nucleotides, with a sugar-phosphate backbone and a nucleobase used as the information-coding-unit. The nucleobases in RNA molecules are: adenine (A), guanine (G), cytosine (C) or uracil (U). The complementary mapping of the nucleobases needs to be slightly updated when the RNA enters the picture. Then, A is complementary to U as well as T. Both DNA and RNA sequences have two ends, one is called the 5'-end and the other one is called the 3'-end. For the purposes of this Thesis it is only important to know that the reading direction of the sequences is from the 5'-end to the 3'-end. Figure 1.2 illustrates a comparison of RNA and DNA.

[1] The fundamental principles of biochemistry are taken from the book [49].

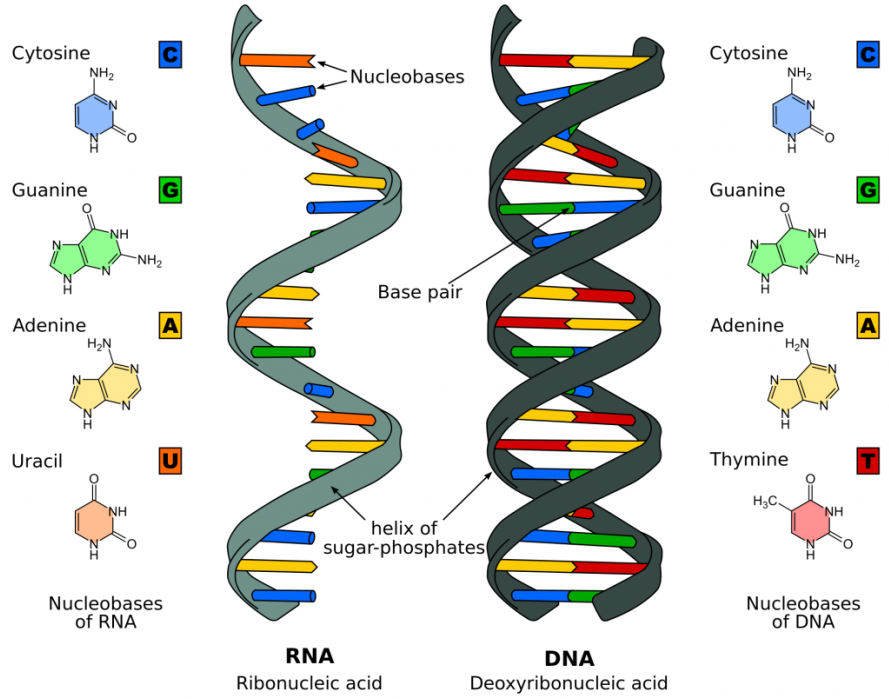


Figure 1.2: This figure demonstrates the difference of the single stranded RNA and DNA double helix. On the margins of the picture, the structures of the nucleobases are presented. (Picture taken from https://teachmephysiology.com/biochemistry/protein-synthesis/dna-transcription)

There are different types of RNA with different tasks. Among these types of RNA, three are present in any living cells: tRNA, rRNA and mRNA. The tRNA (transfer RNA) provides the amino acids for synthesizing proteins. The rRNA is one of the central parts of the ribosome. The mRNA is constructed during the process of transcription and transports the genetic information from the DNA to the ribosome. During the transcription, one snippet of a DNA sequence is prepared for the protein synthesis. This snippet is called a protein coding gene. Such protein coding genes are sections in the DNA which hold the information to construct proteins. These genes are the basic model of the genetic information. After the mRNA is constructed from the protein coding genes, the information is transported to the ribosomal protein factories. In the ribosomes, the translation starts the protein synthesis. Table 1.1 summarizes the structure of DNA and RNA. The next section explains the process of translation in more detail.

	DNA	RNA
Sugar	deoxyribose	ribose
Bases	Adenine (A) Thymine (T) Guanine (G) Cytosine (C)	Adenine (A) Uracil (U) Guanine (G) Cytosine (C)
Complementary Base Pairs	$A \leftrightarrow T$ $C \leftrightarrow G$	$A \leftrightarrow$ U $C \leftrightarrow G$
Structure	two strands, arranged in a double helix	single strand
Function	DNA replicates and stores genetic information.	Among other things, the RNA transports genetic information to ribosomal protein factories.

Table 1.1: Short summery of the structure of DNA and RNA

1.1.2 Translation and protein synthesis

During the translation, the ribosome translates the mRNA into specific *polypeptide chains*. These polypeptide chains then fold into proteins which regulate and control the main purpose of the cell itself. In contrast to prokaryotic cells, eukaryotic cells require the mRNA to be processed before being translated. This is called mRNA processing.

Pre-translational mRNA processing

The mRNA which is the immediate result of the transcription is called *pre-mRNA*. In eukaryiotic cells, this pre-mRNA needs to be processed before it can be translated by the ribosomes. The mechanism of mRNA processing is a composition of three steps, all of which are needed to convert the pre-mRNA into mature mRNA. The three steps include 5' Capping, polyadenylation and splicing and is illustrated in Figure 1.3.

5' Capping Capping describes the addition of a cap to the 5' end of mRNA. The cap stabilizes the immature pre-mRNA and allows the ribosomes to recognize the mature mRNA as such.

Polyadenylation To further stabilize the unstable pre-mRNA a poly(A) tail, *i.e.* a tail consisting of numerous Adenosin bases, is added to the 3' end of mRNA.

Splicing Slicing separates pre-mRNA into introns and extrons. Introns are non-coding sequences which do not appear in the mature mRNA, while exons are coding sequences which compose the mRNA. One pre-mRNA can code numerous proteins by separate splicing processes in which different overlapping extrons are used. (see Figure 1.3)

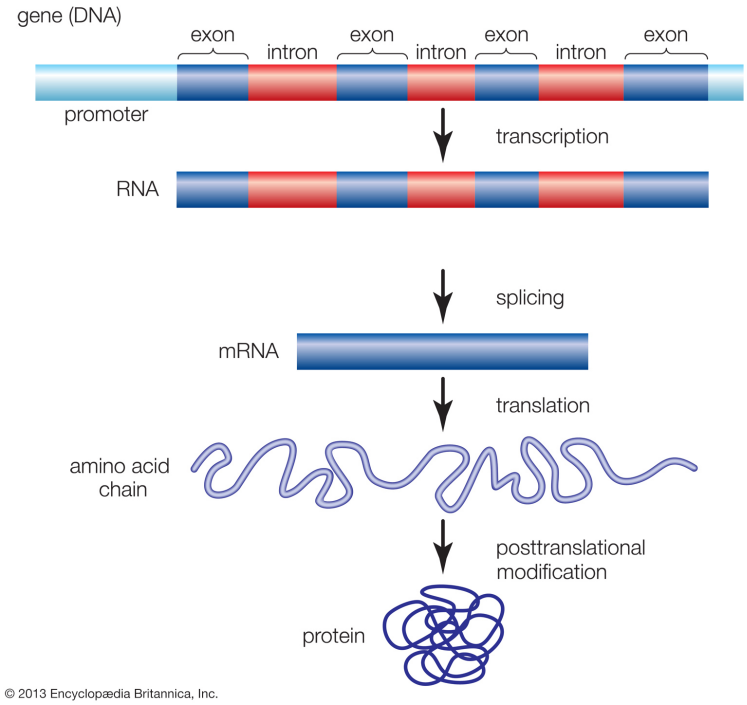


Figure 1.3: Summary of the mRNA processing in a eukaryotic cell. The figure illustrates the splicing process. (Picture taken from https://cdn.britannica.com/96/114896-050-3F22219B/Genes-promoter-regions-production-introns-exons-gene.jpg sorce Encyclopædia Britannica, Inc.)

To build the polypeptide chain encoded in an mRNA, the ribosome reads the mRNA with three nucleotides at a time. This combination of three nucleotides is called a *codon*. The reading is implemented by docking suitable tRNA (transfer RNA) *anti-codons* to the mRNA codons. An anti-codon consists of complementary nucleotides in reversed order. Usually, tRNA is illustrated with three hairpin loops, the D loop, the T loop and the anti-codon loop. Its actual 3D shape is best described as a distinctive L-shape. Figure 1.4 presents an example of the codon anti-codon pairing and shows the schematic illustration of a tRNA.

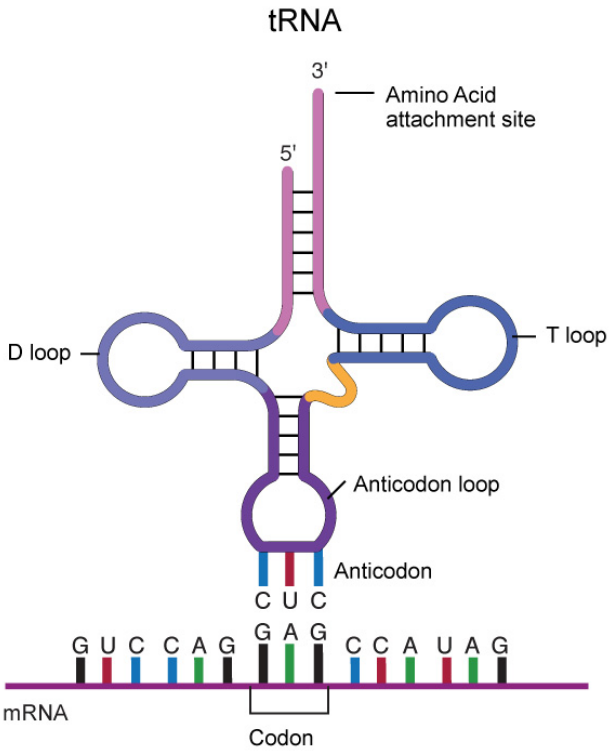


Figure 1.4: A schematic illustration of the tRNA. Additionally, this figure shows how codon (GAG) in the mRNA is paired with its anti-codon (CTC) in the tRNA. (Picture taken from `https://rarediseases.info.nih.gov/GlossaryDescription/474/0`)

The entire process is depicted in a simplified model in Figure 1.5. It also shows the L-shape of the tRNA and the composition of the polypeptide chain.

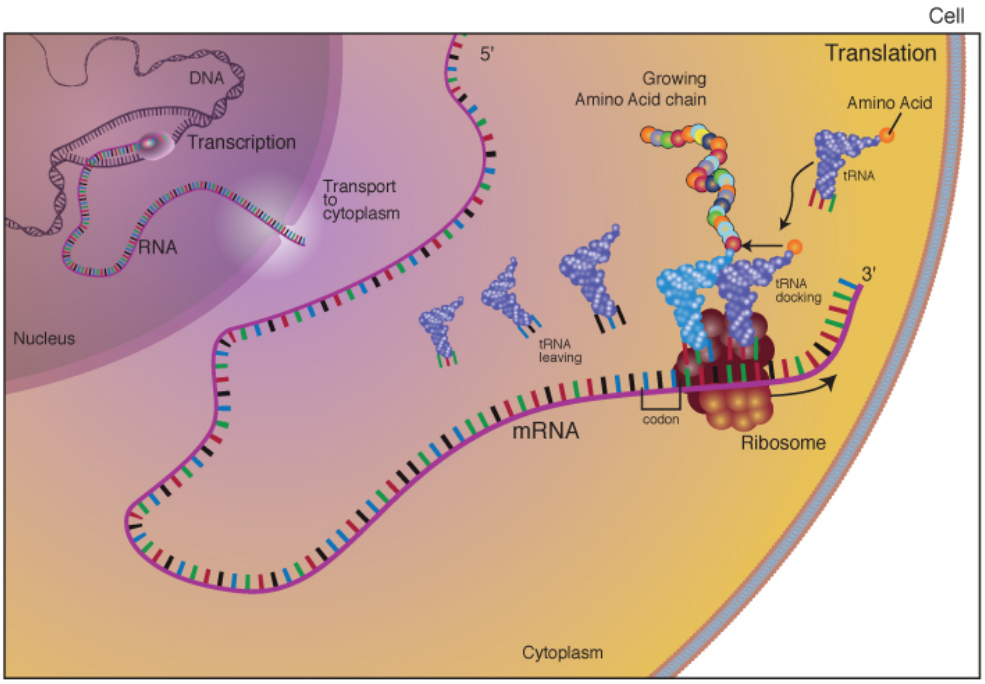


Figure 1.5: The translation process inside a cell: the ribosome uses the tRNA to compose the polypeptide chain. (Picture taken from `https://rarediseases.info.nih.gov/files/glossary/english/translation_lg.jpg` source: National Human Genome Research Institute's Talking Glossary of Genetic Terms)

The process of translation is bounded by a number of rules. These rules apply to almost all living beings. Each translation is initiated by a starting signal. The starting signal marks the position in the mRNA where the ribosome has to begin with the encoding process. The most common start codon is ATG. The translation ends with a so-called stop codon. In the *standard genetic code* (SGC), they are: TAA, TAG and TGA. The SGC is a translation table used to translate codons into amino acids. Even though there are 33 other coding tables besides the SGC, they all differ only slightly from the SGC and have few other codon-to-amino acid translations[2]. However, some of them do not contain a codon that exclusively encodes for the stop signal. The standard genetic code will be described in more detail in the following section.

[2]At the website: `https://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi` all translation tables are listed with their differences to the SGC.

Standard genetic code (SGC)

Almost all living cells use the standard genetic code for the protein synthesis. With the first discovery of the translation of the codon TTT in 1961 by Marshall Nirenberg and Heinrich Matthaei, the encoding of the SGC was imminent [60]. In 1966, almost 5 years later, the entire SGC was encoded. The SGC is a set of rules which serves as a translation table to map the 64 ($4^3 = 64$ all 3-letter words over four nucleobases) codons to the 20 amino acids used in cells and a stop signal. A list of all amino acids is provided in the Appendix in Figure II.1. The translation is fully illustrated in Figure 1.6

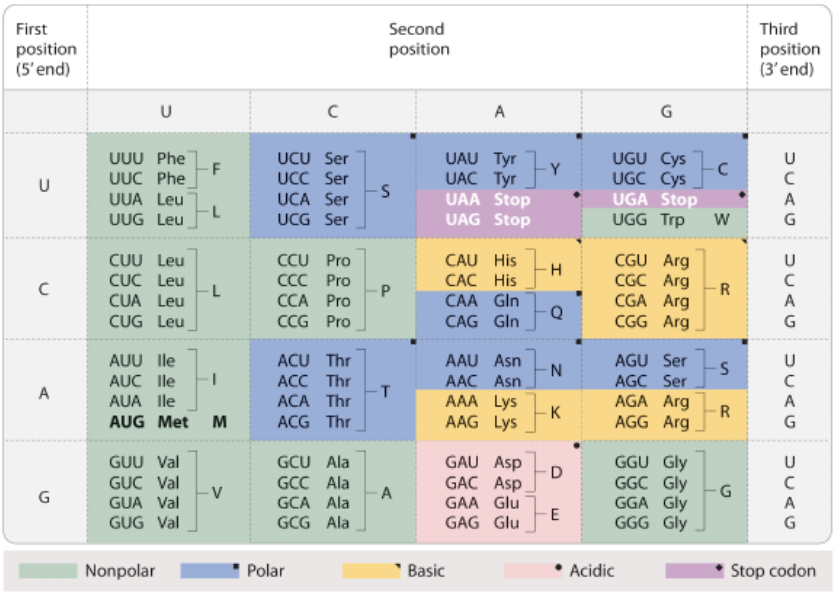


Figure 1.6: The figure depicts all translation rules of the standard genetic code, which specifies the 64 codons for the 20 amino acids. The translation is surjective, meaning that the same amino acid can be encoded by more than one codon but is encoded by at least one. In most cases, the bold codon ATG serves as start codon and the bold codes TAA, TAG and TGA code as the stop signal. (Picture taken from https://www.chegg.com/homework-help/questions-and-answers/standard-genetic-code-shown-table-41-many-codons-amino-acids-allow-synonymous-mutations-th-q40769295)

One of the most significant features of the SGC is its *degeneration*. In this context, degeneration denotes the fact that some amino acids are encoded by more than one codon. For instance the amino acid Ala (Alanine) is encoded by GCT, GCC, GCA and GCG. Similar to Ala, the amino acids Gly (Glycine), Pro (Proline), Thr (Threonine) and Val (Valine) are encoded by four codons. Consequently, all

of these amino acids have a degeneracy of four. A closer observation of the codons coding for one of the four amino acids reveals that the first two bases are the same for each of them. For example, all codons starting with the two bases GC code for the amino acid Ala. This is a mechanism to improve the translation error robustness. Hence, a point mutation of the last base has no influence on the encoded amino acid. Table 1.2 lists all amino acids and their degeneration. The last column of the table with the header *Compressed* shows the range of point mutations for each amino acid which has no effect on the coding. 90% of the amino acids are encoded by at least two codons. Three amino acids Arg (Arginine), Leu (Leucine) and Ser (Serine) even have a degeneracy of six. It is impressive that none of the redundant coding is wasted, in so far that for each codon that codes for an amino acid with a degeneration greater than one there is at least one codon with two common bases that codes for the same amino acid. The origin and purpose of this property is one of the most controversially discussed questions in the evolution of the genetic code, which will be outlined in the following subchapter.

Amino acid & Degeneracy		DNA codons	Compressed
Arg	6	CGT, CGC, CGA, CGG; AGA, AGG	CGN, AGR; or CGY, MGR
Leu	6	CTT, CTC, CTA, CTG; TTA, TTG	CTN, TTR; or CTY, YTR
Ser	6	TCT, TCC, TCA, TCG; AGT, AGC	TCN, AGY
Pro	4	CCT, CCC, CCA, CCG	CCN
Ala	4	GCT, GCC, GCA, GCG	GCN
Val	4	GTT, GTC, GTA, GTG	GTN
Thr	4	ACT, ACC, ACA, ACG	ACN
Gly	4	GGT, GGC, GGA, GGG	GGN
Ile	3	ATT, ATC, ATA	ATH
Asn	2	AAT, AAC	AAY
Lys	2	AAA, AAG	AAR
Asp	2	GAT, GAC	GAY
Phe	2	TTT , TTC	TTY
Cys	2	TGT, TGC	TGY
Gin	2	CAA, CAG	CAR
Glu	2	GAA, GAG	GAR
Tyr	2	TAT, TAC	TAY
His	2	CAT, CAC	CAY
Met	1	ATG	
Trp	1	TGG	
START	1	ATG	
STOP	3	TAA, TGA, TAG	TRA, TAR

Table 1.2: The left column lists all 20 amino acids used for protein synthesizes and their degeneracy. The middle column shows the DNA codes coding for the amino acid. In the last column the possible compressions using the IUPAC notation are given. (The compression column uses the following expressions form the IUPAC notation: $H \in \{A, T, C\}$), $M \in \{A, C\}$, $N \in \{A, T, C, G\}$, $R \in \{A, G\}$ and $Y \in \{C, T\}$

1.1.3 Evolution of the genetic code

Almost every living organism uses the standard genetic code (SGC) to translate 64 codons into 20 amino acids and the stop signal. Despite the few exceptions of the SGC, this observation offers considerable insights into evolution of the genetic code. However, to date, all existing theories on the evolution of the genetic code

are controversial. If we include the 33 divergent tables in relation to the SGC, we can sketch the evolution as shown in Figure 1.7.

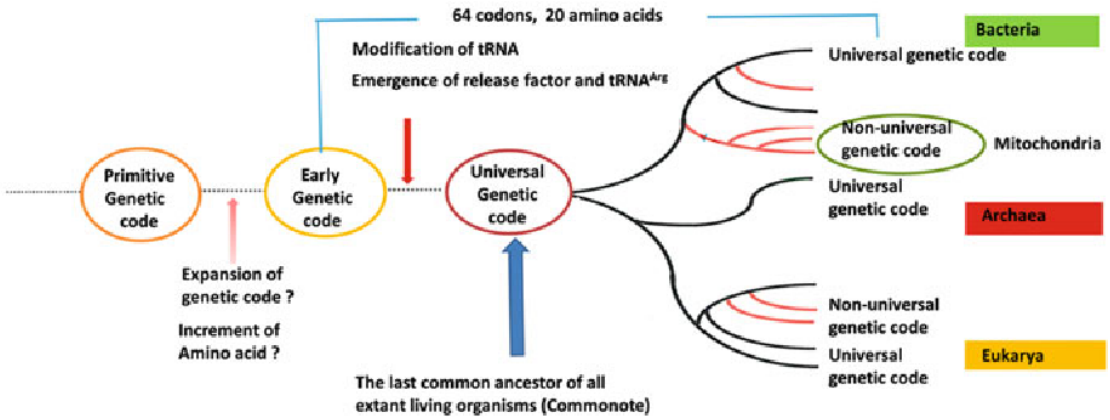


Figure 1.7: A schematic presentation of the microbiological evolution introduced in [74]. The evolution line marks the important milestones of the evolution. Starting with a primitive code which first evolved to the early genetic code and then to the LUCA (last common ancestor code). After LUCA, the branches of evolution divide into the forms of life we have today. (Picture taken from [74])

In the so-called common descent theories, which is currently assessed by the academic community as the most plausible theory, it is postulated that all living beings can be traced back to one common ancestor code [33]. The code in which the branches of the evolutionary tree separate is referred to as LUCA (last common ancestor code), while a pre-LUCA genetic code is referred to as primitive genetic code [74, 47]. The translation mechanisms of the ordinary cells from which modern life arose are comparable to those of modern cells but far more primitive. Nevertheless, the LUCA cells, which represented an important milestone in evolution, already had a complex protein synthesis system. It is the general consensus of researchers that such a system as LUCA cannot arise spontaneously. Consequently, it can be assumed that LUCA is just one of many steps in the evolution of the genetic code.

In contrast to the common knowledge at that time, in 1968, Crick published [11] the frozen accident theory which states that the genetic code was generated by chance and has remained frozen ever since. According to this theory, a change in the code would be harmful to its functionality and its conservation would have to be the aim of nature. Based on this scenario, Crick added the theory that the early code system is based on RNA only. However, this has been refuted by the fact that RNA cannot carry out these tasks [6]. New approaches as in the article [8, 65] adopted Crick's theory of an RNA based code system and state that: "A molecular

replicator with two components –RNA and peptide – overcomes these problems and may be a better fit." [65].

Apart from the frozen accident theory, other theories claim to explain the origin of the genetic code. The most common ones are: the *stereochemical* theory [64, 79], the *adaptive* theory [76, 35] and the *coevolution* theory [77]. These theories are briefly outlined in the following paragraphs.

The stereochemical theory claims that there is a stereochemical attraction between codons or rather anti-codons and their assigned amino acids [64, 79]. In comparison to this theory, the coevolution theory focuses on the idea that the amino acids advanced biosynthetically from a set of precursor amino acids. This set of precursor amino acids goes back to Crick's article [11] and was according to the theory initially coded while the rest evolved through biosynthetic pathways [77]. A characterization of the precursor amino acids by Higgs in the article [44] suggests that the amino acids Gly, Ala, Asp, Glu and Val were the first amino acids in the polypeptide coding. These precursor amino acids consist exclusively of the amino acids encoded by a code with a leading base G. In Higgs hypercritical evolution, the code evolved from a four column code where all codons in one column encode for the same amino acid: NUN = Val, NCN = Ala, NAN = Asp and/or Glu, and NGN = Gly.

The adaptive theory is derived from the degeneracy of the SGC. It postulates that evolution aimed to obtain a code that minimizes errors through mutations [76, 35]. According to the article [61], the standard code is the result of a partial optimization of a random code for robustness against translation errors. The authors interpret their results as such that the compromise between the increasing robustness against translation errors and the extension of the coding table could be the reason for the code not being fully optimized. They also found that the evolution of the code can, thus, be represented as a combination of adaptation and frozen accident.

Although all four models attempt to explain the evolution of the genetic code from different perspectives, it is likely that they all play a crucial role in the evolution of the genetic code. A rather less controversial theory is that life based on LUCA dates back 650 million years. This is due to the identification of 6331 groups of genes common to all living creatures. Hence, e.g. 55% of human protein-coding genes belong to gene groups that were present in the hypothetical first animals [73, 62].

1.2 Code theory

The degeneracy of the SGC (standard genetic code) is the foundation of the adaptive theory. In addition to this feature, which guarantees a certain robustness

against mutation errors during translation, the theory of the existence of a block code in genetic sequences was inspired by Watson and Crick when they published their article in 1957 [12]. This new branch of mathematical biology elaborates on the idea that a block code, *i.e.* codes consisting of words of a fixed length over an arbitrarily finite alphabet, assists the ribosome to avoid reading errors. In their work, Watson and Crick suggest that a comma-free code would ensure reading in the correct frame, *i.e.* a code which can be read only in the correct frame without a separating symbol like a comma or white space.

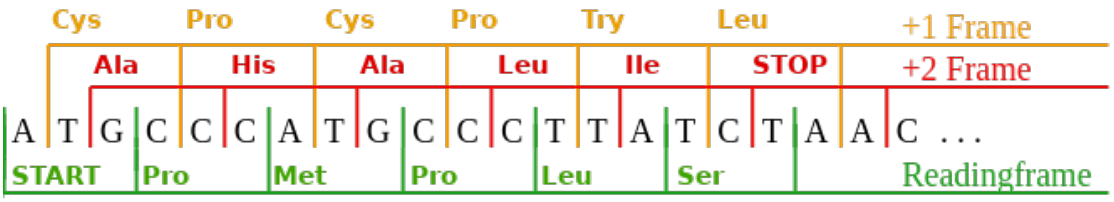


Figure 1.8: A brief illustration of the three reading-frames in the genetic code. The green reading-frame is the correct one.

This idea was further enhanced by the discovery of Golomb et al. in 1958 that a comma-free code over an alphabet of four letters with a word length of three has a maximum size of 20 elements [36] which is exactly the number of amino acids used for the protein synthesis. Although these facts supported the comma-free hypothesis, the codon TTT which was excluded from the comma-free codes was found in coding sequences in 1961, thus, disproving the hypothesis [60]. Even if the comma-free codes do not correspond to the modern SGC anymore, they could be a step in the evolutionary process+[59]. This hypothetical model is supported by new evidence of a circular code: a weakened version of comma-free codes which might still have influence on the translation process [1, 56, 50, 51, 70, 54]. In the subsequent section, the mathematical basics of block codes with a main focus on circular codes and comma-free codes are summarized. Additionally, new mathematical methods from graph theory and group theory to work with these block codes are presented[3].

[3]The fundamental principles of the word theory and the coding theory are taken from the book [5].

1.2.1 Definitions and notations

The following section summarizes the mathematical definitions and notations used in this thesis. In addition to this section, we provide a list of the most important notations and operators at the end of the thesis.

The notation $\mathcal{X} \subset \Sigma^\ell$ defines a code $\mathcal{X}$ which is a subset of the set of all ℓ-letter words over the alphabet Σ. We define Σ to be an alphabet of cardinality of at least 2. The cardinality $|\Sigma|$ of Σ is denoted as $n \coloneqq |\Sigma|$. For the set of all finite words with letters in Σ, we use the standard word-theory notation Σ^*.The Set Σ^* also includes the empty word denoted as ϵ. The set of all finite words with letters in Σ without the empty word ϵ is denoted as Σ^+. Hence, $\Sigma^\ell \subset \Sigma^+ \subset \Sigma^*$ for any finite $\ell \in \mathbb{N}$.

Example 0.1. *Let $\Sigma = \{0, 1\}$ be the binary alphabet and $\ell = 3$. Hence, Σ^ℓ are all binary 3-letter words.*

$$\Sigma^\ell = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

Let us assume that $\mathcal{X} = \{000, 111\}$, then it follows that $\mathcal{X} \subset \Sigma^\ell$

Most codes in this thesis will be codes over the genetic alphabet, which has cardinality four. We set $\mathfrak{B} \coloneqq \{A, T, G, C\}$. The *complementary mapping* of the nitrogenous bases is denoted as the mapping function $c(\cdot)$:

$$c(T) \coloneqq A;\ c(C) \coloneqq G \text{ and vice versa}$$

We define the function $c(\cdot)$ so that for any code $\mathcal{X} = \{w_1, w_2, \ldots, w_x\}$ of size $x \in \mathbb{N}$ the following holds: $c(\mathcal{X}) \coloneqq \{c(w_1), c(w_2), \ldots, c(w_x)\}$. For the *reversing permutation* of a word $w = b_1 \ldots b_\ell$ we use the following notation:

$$\overleftarrow{b_1 \ldots b_\ell} \coloneqq b_\ell \ldots b_1$$

As above the reversing permutation is defined so that: $\overleftarrow{\mathcal{X}} \coloneqq \{\overleftarrow{w_1}, \overleftarrow{w_2}, \ldots, \overleftarrow{w_x}\}$.

Another permutation used in this thesis is the *circular permutation*. Let $w = b_1 \ldots b_\ell \in \Sigma^\ell$ be a word for some $\ell \in \mathbb{N}$. Then for every $j \in \{0, \ldots, \ell - 1\}$, the *circular j-permutation* (denoted as $\alpha_j(w)$) of w is the word $b_{j+1} \ldots b_\ell b_1 \ldots b_j$.

$$\alpha_j(w) \coloneqq b_{j+1} \ldots b_\ell b_1 \ldots b_j$$

(In particular, the circular 0-permutation of w is w itself.) A word w' is a *circular permutation* of w if w' is the circular j-permutation of w for some $j \in \{0, \ldots, \ell - 1\}$.

We denote $\alpha_j(w)$ as the circular j-permutation of the word w. $\alpha_j(\cdot)$ is defined so that for a code $\mathcal{X} = \{w_1, \ldots, w_x\}$ of size x the following holds: $\alpha_j(\mathcal{X}) = \{\alpha_j(w_1), \ldots, \alpha_j(w_x)\}$

Next, we briefly explain the Cartesian product. Let $A = \{00, 11\}$ and $B = \{01, 10\}$ be simple binary codes of size two. Then $A \times B \coloneqq \{0010, 0001, 1110, 1101\}$ is a new set of words, where each word is a concatenation of a and b, where $a \in A$ and $b \in B$. To denote the x-ary Cartesian power of a code $\mathcal{X}$, we use the word theory standard $\mathcal{X}^x$, *i.e.* $\mathcal{X}^x$ is the Cartesian product of $\mathcal{X} \times \mathcal{X} \times \cdots \times \mathcal{X}$.

Finally, we introduce the principles of group theory. A group is a tuple $(G, \circ)$ where G is a set and $\circ$ is a binary operator. If a tuple $(G, \circ)$ is qualified as a group, then the following must hold:

- For any $a, b \in G$ the combination $a \circ b \in G$.
- There must be an *identity element* $e \in G$ so that for each $a \in G$ it follows that $e \circ a = a \circ e = a$.
- For any $a \in G$ the *inverse* $a^{-1} \circ b \in G$ so that $a \circ a^{-1} = e$.
- The group must be *associativity*, *i.e.* for any $a, b, c \in G$, $(a \circ b) \circ c = a \circ (b \circ c)$

More specific definitions will be introduced in the subsequent sections. In the next section, we introduce the natural properties of codes.

1.2.2 What is a code?

The Cambridge dictionary explains a code in the following words:

> "a system of words, letters, or signs used to represent a message in secret form, or a system of numbers, letters, or signals used to represent something in a shorter or more convenient form"
> → [66]

In more mathematical terms, a code is a set of (code)words so that if an arbitrary message is coded by these (code)words it must be uniquely decodable into the original message. Different codes for different problems have been developed over the time. Some codes have the purpose to hide the coded message, others serve as an adapter between two receivers. Famous representatives of these codes are the Enigma, an encryption device which was used during World War II [48], and computer languages that translate human-readable code into binary commands that can be executed (*i.e.* are understood) by machines. Another domain of codes was developed to maintain and secure the integrity of a message for the receiver. These error-detecting codes are the key tool of this thesis. This thesis aims to bring forward using codes as tool to understanding and thereby, possibly explaining the error robustness in the genetic translation. Definition 1 introduces a formal mathematical definition of codes.

Definition 1. *Let Σ be a finite alphabet and $\mathcal{X} \subseteq \Sigma^\ell$ for some $\ell \in \mathbb{N}$.*

- *For $w \in \Sigma^*$, an* $\mathcal{X}$-decomposition *of w is a word $w_1 \cdots w_r \in \mathcal{X}^r$ with $r \in \mathbb{N}$ such that $w = w_1 \cdot w_2 \cdots w_r$.*
- *A set $\mathcal{X} \subseteq \Sigma^*$ is a code if each word $w \in \mathcal{X}^j$ has a single $\mathcal{X}$-decomposition for any $j \geq 2 \in \mathbb{N}$.*
- *For an integer $\ell \geq 2$, an* ℓ-letter code *is a block code contained in Σ^ℓ.*

The following section introduces circular codes. These error-detecting codes belong to the block codes and allow to retrieve the correct reading-frame in a message. Three examples of a code and two non-codes are given below.

Example 1.1. *Let $\mathcal{X}_2 = \{10, 01, 1101\}$ be a set of binary words. All sequence of composed words in $\mathcal{X}_2$ are uniquely decodable. Hence, $\mathcal{X}_2$ is code.*

Example 1.2. *Let $\mathcal{X}_1 = \{10, 01, 1001\}$ be a set of binary words. Then, the sequence $1001 = 10 \cdot 01$ is not uniquely decodable. Hence, $\mathcal{X}_1$ is not a code*

Example 1.3. *Let $L \subset \{A - Z, a - z\}^+$ be the English language and $\mathcal{X}_3 \subset L$ so that $\mathcal{X}_3 = \{counter, clock, wise\}$. If L was a code then $\mathcal{X}_3^3 \cap L = \emptyset$. Yet, the word counterclockwise is in $\mathcal{X}_3^3$ and L. Therefore $\mathcal{X}_3$ is a code, but L is not*[4].

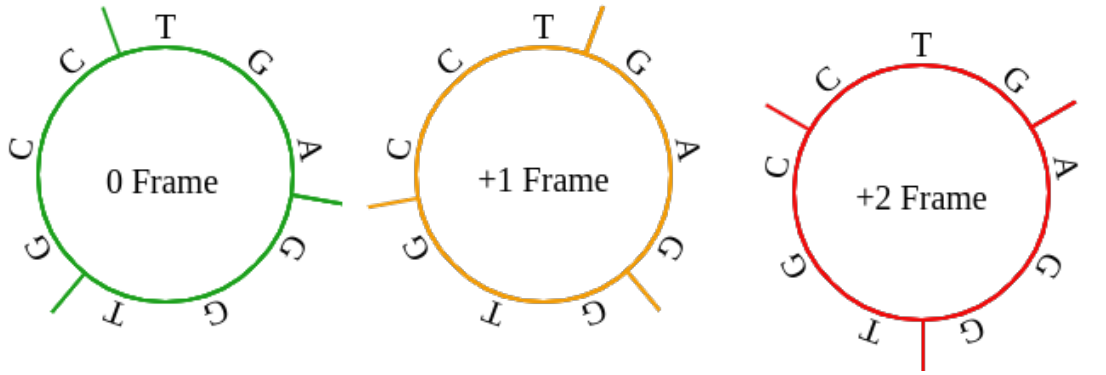
1.2.3 Circular codes

Error-correction codes like circular codes belong to the block code family. Although not all circular codes are block codes, we use them exclusively as such. Block codes are codes which encode data in blocks or words of equal length. This is similar to the process of RNA translation. This leads to assume that block codes play a role in the translation process of trinucleotides. Another argument supporting this hypothesis relates to the main feature of circular codes: the frameshift retrieval. A frameshift describes a reading error of a sequence where you slide from one reading-frame (see Figure 1.8) to another reading-frame, e.g. by skipping a letter. It addresses an open question of the processing of genetic information. Before presenting the biological application of circular codes, we define these codes and explain how they operate in the following.

A circular code is defined so that any concatenation of words of the code written on a cycle can only be decomposed into words of the code in one reading-frame. Let $\mathcal{X} = \{TGA, GGT, GCC\} \subset \mathfrak{B}^3$ be a trinucleotide circular code. Since $\mathcal{X}$ is

[4]This is discussed controversially in academia because it can be argued that white space separation is part of the composition process of languages. Therefore, language can be considered a code.

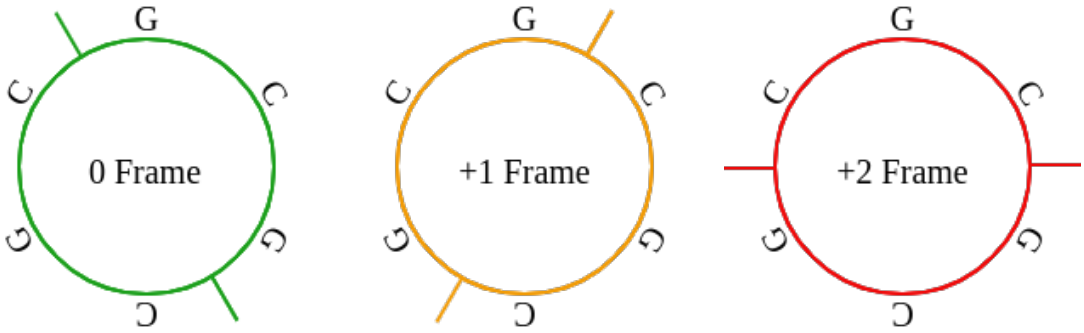
circular, any concatenation of the three words can only be decomposed into these three words in one reading-frame. To demonstrate this, the Figure 1.9 illustrates the decomposition of the sequence $TGAGGTGCC$ written on a cycle. While the three words in the 0 frame are in the code $\mathcal{X}$, the words in the +1 frame and the words in the +2 frame are not in $\mathcal{X}$.



Readingframe: $TGA{\cdot}GGT{\cdot}GCC$ +1 Frame: $GAG{\cdot}GTG{\cdot}CCT$ +2 Frame: $AGG{\cdot}TGC{\cdot}CTG$

Figure 1.9: An illustration of the three reading-frames of the sequence $TGA \cdot GGT \cdot GCC$.

The next example presents a sequence of words in the trinucleotide code $\mathcal{X} = \{TGA, CGC, GCG\} \subset \mathfrak{B}^3$, which is a non-circular code. The proof is illustrated in Figure 1.10. It is shown that the sequence $GCGCGC$ written on a cycle is decomposed in all three frames into words of $\mathcal{X}$.



Readingframe: $GCG{\cdot}CGC$ +1 Frame: $CGC{\cdot}GCG$ +2 Frame: $GCG{\cdot}CGC$

Figure 1.10: An illustration of the three reading-frames of the sequence $GCG{\cdot}CGC$. All three frames are decomposed into the same words.

In the following, we provide a formal definition of the circular codes. This definition uses the $\mathcal{X}$-decomposition definition which is defined in Definition 1 of a code.

Definition 2. *Let $\mathcal{X} \subseteq \Sigma^\ell$ be an ℓ-letter code.*

- *Let m be a positive integer and let $w_1 \dots w_m \in \mathcal{X}^m$. A* circular $\mathcal{X}$-decomposition *of the concatenation $w \coloneqq w_1 \cdots w_m$ is an $\mathcal{X}$-decomposition of a circular permutation of $\alpha_j(w)$ for some $j \in \{1, \dots, m-1\}$.*
- *The code $\mathcal{X}$ is* circular *if for every $m \in \mathbb{N}$ and every word $w \coloneqq w_1 \dots w_m \in \mathcal{X}^m$, the w admits a unique circular $\mathcal{X}$-decomposition.*

Next, we proceed with the definition of *circular permutation classes*. These definitions use the group theory which was introduced in section 1.2.1.

Permutation groups

Let us first denote the *symmetry group* $(S_\ell, \circ)$ where ℓ is a positive integer. This symmetry group S_ℓ acts on the indices of the ℓ-letter words from Σ^ℓ. The group $(S_\ell, \circ)$ is endowed with the group operation of composition. The set of permutations is formally defined as:

$$S_\ell \coloneqq \{g : (1, 2, \dots, \ell) \to (1, 2, \dots, \ell) : g \textit{ is bijective}\}$$

Each $g \in S_\ell$ is predefined and permutes the letters in an ℓ-letter word. For instance, let $\ell = 3$, then

$$S_3 \coloneqq \{(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)\}.$$

If, for instance the permutation $g = (3, 2, 1) \in S_3$ acts on a word $b_1b_2b_3 \in \Sigma^3$, it follows that $g(b_1b_2b_3) = b_3b_2b_1$.

One important subgroup $\mathcal{A}_\ell$ of S_ℓ is called the *circular permutations group* $\mathcal{A}_\ell \subset S_\ell$. It is defined as follows. Let Σ^ℓ be a set of all ℓ-letter words under an alphabet Σ. Further, we denote $(\mathcal{A}_\ell, \circ)$ to be the circular permutations group. Where

$$\mathcal{A}_\ell \coloneqq \{\alpha_0 \coloneqq (1, 2, \dots, \ell), \alpha_1 \coloneqq (2, \dots, \ell, 1), \dots, \alpha_{\ell-1} \coloneqq (\ell, 1, 2, \dots, \ell-1)\}$$

and $\circ$ denotes the composition of permutations. Be $\alpha_0(\cdot)$ the identity element. Then, $\mathcal{A}_\ell$ is a group acting on Σ^ℓ:

$$\alpha \colon \mathcal{A}_\ell \times \Sigma^\ell \to \Sigma^\ell$$

Next we formally define the circular permutations classes by the means of $\mathcal{A}_\ell$.

Definition 3. *Let ℓ be a positive integer and $\Sigma^\ell \subset S_\ell$ be a set of all words under an alphabet Σ. All words in Σ^ℓ can be separated into so-called circular permutation classes by the means of $\mathcal{A}_\ell$. We define a circular permutation class as an orbit of $\mathcal{A}_\ell$ acting on a word $w \in \Sigma^\ell$.*

$$\mathcal{A}_\ell \cdot w \coloneqq \{\alpha(w) \mid \alpha \in \mathcal{A}_\ell\}$$

The maximum size of an orbit, and therefore a circular permutation class, is equal to the cardinality of $\mathcal{A}_\ell$ which is ℓ. Hence, we donate a circular permutation class that has ℓ members as a complete circular permutation class. The other circular permutation classes are denoted as incomplete circular permutation classes.

Next, we provide an example for Definition 3. In Example 3.1 we demonstrate one circular permutation class of a 3-letter word over an arbitrary alphabet.

Example 3.1. *Let $b_1b_2b_3 \in \Sigma^3$ be a 3-letter word over an alphabet Σ, and $(\mathcal{A}_3, \circ)$ is a group where $\mathcal{A}_\ell \coloneqq \{\alpha_0, \alpha_1, \alpha_2\}$. Then the circular permutation class of $b_1b_2b_3$ is defined as:*

$$\{b_1b_2b_3, b_2b_3b_1, b_3b_1b_2\} = \{\alpha(b_1b_2b_3) \mid \alpha \in \mathcal{A}_\ell\}$$

The set of trinucleotide words can be classified into 20 complete circular permutation classes and four incomplete circular permutation classes. Next, we list all complete circular trinucleotide permutation classes:

$\{AAC, ACA, CAA\}$, $\{AAG, AGA, GAA\}$, $\{AAT, ATA, TAA\}$, $\{ACC, CCA, CAC\}$, $\{ACG, CGA, GAC\}$, $\{ACT, CTA, TAC\}$, $\{AGC, GCA, CAG\}$, $\{AGG, GGA, GAG\}$, $\{AGT, GTA, TAG\}$, $\{ATC, TCA, CAT\}$, $\{ATG, TGA, GAT\}$, $\{ATT, TTA, TAT\}$, $\{CCG, CGC, GCC\}$, $\{CCT, CTC, TCC\}$, $\{CGG, GGC, GCG\}$, $\{CGT, GTC, TCG\}$, $\{CTG, TGC, GCT\}$, $\{CTT, TTC, TCT\}$, $\{GGT, GTG, TGG\}$, $\{GTT, TTG, TGT\}$.

and all trinucleotide incomplete circular permutation classes:

$\{AAA\}$, $\{TTT\}$,$\{CCC\}$,$\{GGG\}$.

Every word w in an incomplete circular permutation class is characterized by the condition that there must be at least one $\alpha \in \mathcal{A}_\ell$ so that $\alpha(w) = w$. Thus, if a code contains such a word, it cannot be circular.

Figure 1.11 shows how the circular trinucleotide code retrieves the reading-frame in an RNA sequence.

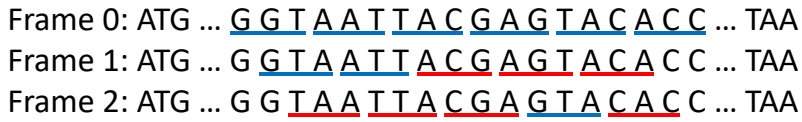


Figure 1.11: Readingframe retrieval in genes with a trinucleotide circular code $\mathcal{X} = \{AAC, AAT, ACC, ATC, ATT, CAG, CTC, CTG, GAA, GAC, GAG, GAT, GCC, GGC, GGT, GTA, GTC, GTT, TAC, TTC\}$ identified in genes. A frameshift is detected after no more than 15 nucleotides. The codons underlined in blue belong to $\mathcal{X}$. The trinucleotides underlined in red do not belong to $\mathcal{X}$.

A circular code excludes all incomplete circular permutation classes (see 3). In the domain of trinucleotide codes, theses are the so called identity codons: $\{AAA, CCC, GGG, TTT\}$. If a code is circular, it also follows that it contains no more than one word from each complete circular permutation class. It has also been shown that the number of complete circular permutation classes under an arbitrary alphabet Σ with a word length of ℓ is equal to the maximum size of a circular code under Σ^ℓ [23]. Thus, the maximum size of a trinucleotide code is 20.

1.2.4 Comma-free codes

Although Crick's suggestion that the genetic code consists of a comma-free code [12] has been disproved, these codes still play a role in hypothetical evolutionary theories such as those presented in Section 1.1.3. Like the circular code, the comma-free codes belong to the block codes. The comma-free codes are a more restrictive version of the circular codes. A code is called comma-free if any concatenation of words contain only words of the code in one reading-frame. Such codes immediately detect a reading-frameshift. Definition 4 illustrates a formal notation of the comma-free codes.

Definition 4. *Let $\mathcal{X} \subseteq \Sigma^\ell$ be an ℓ-letter code. $\mathcal{X}$ is comma-free if*

$$\mathcal{X}^2 \cap (\Sigma^+ \times \mathcal{X} \times \Sigma^+) = \emptyset$$

Example 4.1 shows a non comma-free code.

Example 4.1. *Let $L \subset \{A-Z, a-z\}^+$ be the English language and $\mathcal{X} \subset L$ so that $\mathcal{X} = \{spanking, kinglike, timespan\}$. Then the sequence **timespankinglike** $\in \mathcal{X}^2$ shows that $\mathcal{X}$ and thus L cannot be comma-free. The circular 5-permutation of the sequence **timespankinglike** reveals the word "spanking", which is also a word in $\mathcal{X}$.*

timespan, kinglike or time,spanking,like

Hence, it does not meet the definition of comma-free codes.

The next Example 4.2 illustrates an improved comma-free version of the code in Example 4.1.

Example 4.2. *Let $L \subset \{A-Z, a-z\}^+$ be the English language and $\mathcal{X} \subset L$ so that $\mathcal{X} = \{spanking, gniklike, timespan\}$. This small adaptation of Example 4.1 converts $\mathcal{X}$ into a comma-free code.*

Figure 1.12 illustrates how a comma-free trinucleotide code retrieves the reading-frame in an RNA sequence.

Frame 0: ATG … A G A C G A T T A G C C T C A A C A … TAA
Frame 1: ATG … A G A C G A T T A G C C T C A A C A … TAA
Frame 2: ATG … A G A C G A T T A G C C T C A A C A … TAA

Figure 1.12: Readingframe retrieval in genes with the comma-free code $\mathcal{X} = \{ACA, AGA, CGA, GCC, TCA, TTA\}$. A frameshift is detected immediately, after no more than three nucleotides. The trinucleotides (words of length 3) underlined in blue belong to $\mathcal{X}$, the trinucleotides underlined in red do not belong to $\mathcal{X}$.

1.2.5 Additional properties of codes

Circular codes and comma-free codes can satisfy additional properties. These properties are: (1) maximal as the code cannot be contained in a circular/comma-free code of higher cardinality, (2) self-complementary, since for any codon in the code, the associated anti-codon is also in the code and (3) C^n, *i.e.* all circular permutations of the code are also circular codes. Another property which only relates to comma-free codes are the so called (4) strong comma-free codes. These codes are defined so that any prefix of a word in such a code cannot be the suffix of a word in the code and vice versa.

Definition 5. *Let $\mathcal{X} \subset \Sigma^\ell$ be a circular or even comma-free code over an arbitrary alphabet Σ with a word length ℓ. The code is called maximal if $|\mathcal{X}|$ is equal to the number of complete circular permutation classes in Σ^ℓ [23].*

A trinucleotide circular code is defined maximal if it contains 20 codons. Definition 6 presents the definition of a self-complementary code.

Definition 6. *Let $\mathcal{X} \subset \mathfrak{B}^*$ be a code over the alphabet of nitrogenous bases. If $\mathcal{X}$ is a self-complementary code, then, for any word $w \in \mathcal{X}$, it contains the associated anti-word $c(\overleftarrow{w}) \in \mathcal{X}$. The anti-word of a word is the reversed complementary word $c(\overleftarrow{w})$. Hence, $\mathcal{X}$ is self-complementary if:*

$$\mathcal{X} = c(\overleftarrow{\mathcal{X}})$$

Example 6.1. *Let* $\mathcal{X} = \{ACG, CGT, TTC, GAA\}$ *be a trinucleotide code. Then the code* $\mathcal{X}$ *is self-complementary, since* $c(\overleftarrow{ACG}) = CGT$ *and* $c(\overleftarrow{TTC}) = GAA$. *Hence,*

$$\mathcal{X} = \{c(\overleftarrow{CGT}), c(\overleftarrow{ACG}), c(\overleftarrow{GAA}), c(\overleftarrow{TTC})\}.$$

A maximal code of size 20 can even be self-complementary or C^3. Let us introduce C^ℓ property next.

Definition 7. *Let* $\mathcal{X} \subset \Sigma^\ell$ *be a circular or even comma-free code over an arbitrary alphabet* Σ *with a word length* ℓ. *The code is called* C^ℓ *if all circular permutations of* $\mathcal{X}$ *are, again, circular codes. The code* $\mathcal{X}$ *is a* C^ℓ *code if and only if all codes* $\alpha_j(\mathcal{X})$ *for any* $j \in \{0, \ldots, \ell - 1\}$ *are circular codes.*

Compared to a C^3 code, a strong comma-free code cannot contain 20 codons. Such a code is considered maximal if it contains 9 codons [26]. This supports the theory that the increasing robustness against translation errors, and the extension of the coding table could be the reason for the code not being fully optimized [61].

Definition 8. *Let* $\mathcal{X} \subseteq \Sigma^\ell$ *be an* ℓ*-letter code.* $\mathcal{X}$ *is strong comma-free if:*

$$(\mathcal{X}^2 \cap (\Sigma^+ \times \mathcal{X})) \cup (\mathcal{X}^2 \cap (\mathcal{X} \times \Sigma^+)) = \emptyset$$

1.2.6 Representing graph

Recently, a new graph-theoretical approach to the study of circular codes (see definition 9) was introduced [27]. An original approach to 2-letter words [2] was first generalized for the genetic alphabet [27] and later extended to words with any letter over any finite alphabet [25]. The approach uses a directed graph, *i.e.* a set of vertices or nodes connected by edges or arcs. In a directed graph, these edges or arcs are directed, *i.e.* they only go from one vertex to another and not back. The graph is constructed so that an ℓ-letter code is circular if and only if the corresponding graph is acyclic. This breakthrough allows to reduce the problem of deciding whether a code is circular or not to the problem of whether a graph is acyclic or not. In the following, we define the graph to be associated with a code.

Definition 9. *Let* $\ell \in \mathbb{N}$, *and let* $\mathcal{X} \subseteq \Sigma^\ell$ *be an finite* ℓ*-letter block code. A* $\mathcal{G}(\mathcal{X}) = (V(\mathcal{X}), E(\mathcal{X}))$ *is defined with a set of vertices* $V(\mathcal{X})$ *and a set of ARCS* $E(\mathcal{X})$. *Let us define the graph* $\mathcal{G}$ *associated with a code* $\mathcal{X}$.

- $V(\mathcal{X}) \coloneqq \{b_1 \cdots b_i, b_{i+1} \cdots b_\ell \,:\, b_1 \cdots b_\ell \in \mathcal{X}$ *for every* $i \in \{1, ..., \ell - 1\}\}$
- $E(\mathcal{X}) \coloneqq \{w_1 \to w_2 \,:\, w_1 \cdot w_2 \in \mathcal{X}$ *and* $(w_1, w_2) \in V(\mathcal{X})\}$

The graph $\mathcal{G}(\mathcal{X})$ *is the graph* associated *with* $\mathcal{X}$. *For each* $i \in \{1, \ldots, \ell\}$, *the vertices of* $\mathcal{G}(\mathcal{X})$ *that correspond with words of length* i *are referred to as* i-nodes.

Example 9.1. *Let* $\mathcal{X} = \{ACC,\ GAG,\ CCG\}$ *be a a circular trinucleotide code. Then Figure 1.13 shows the graph* $\mathcal{G}(\mathcal{X})$ *associated with* $\mathcal{X}$.

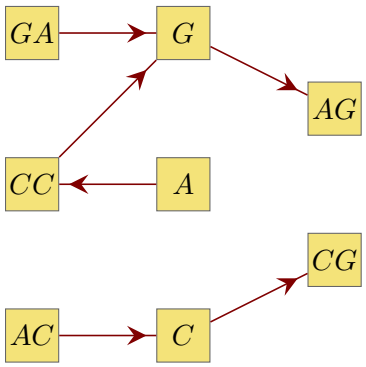


Figure 1.13: $\mathcal{G}(\mathcal{X})$ associated with the code $\mathcal{X} = \{ACC,\ GAG,\ CCG\}$

In the article [27] some important theorems related to a graph and the associated code were illustrated. These theorems extend the means to recognize properties of a code by the associated graph. The following theorems, that were already put forward earlier are also explained below in Theorem 1.2.1, Theorem 1.2.2 and Theorem 1.2.3 which follow subsequently.

Theorem 1.2.1. *(Theorem 2.6, [27]). Given a trinucleotide code* $\mathcal{X} \in \mathfrak{B}^3$, *the following statements are equivalent:*

1. $\mathcal{X}$ *is circular;*

2. $\mathcal{G}(\mathcal{X})$ *is acyclic.*

Theorem 1.2.1 introduces one of the major breakthroughs in the research of circular codes. Before Theorem 1.2.1, in order to conclude whether a code is circular, one needed to check if any concatenation of infinite words of a code written on a cycle can only be decomposed in words of the code in one reading-frame. Consequently, this makes it an undecidable problem. With the discovery of this theorem 1.2.1, this problem became decidable.

Theorem 1.2.2. *(Theorem 2.11, [27]). Let* ℓ *be a positive integer and* $\mathcal{X} \in \Sigma^\ell$ *be a* ℓ*-letter code over an arbitrary alphabet, then the following statements are equivalent:*

1. *$\mathcal{X}$ is comma-free;*

2. *The length of the longest path in $\mathcal{G}(\mathcal{X})$ is 2 at the most.*

Theorem 1.2.2 and Theorem 1.2.3 facilitated the definition of a code as comma-free or even strong comma-free.

Theorem 1.2.3. *(Theorem 2.7, [27]). Let ℓ be a positive integer and $\mathcal{X} \in \Sigma^{\ell}$ be a ℓ-letter code over an arbitrary alphabet. Then the following statements are equivalent:*

1. *$\mathcal{X}$ is strong comma-free;*

2. *The length of the longest path in $\mathcal{G}(\mathcal{X})$ is 1 at the most.*

Representing graph components

For each edge e in $\mathcal{G}(\mathcal{X})$, there is $i \in \{1, \ldots, \ell - 1\}$, so that e goes from a i-node to a $(\ell - i)$-node. For each $i \in \{1, \ldots, \lfloor \ell/2 \rfloor\}$, the *$i$-component* of $\mathcal{G}(\mathcal{X})$ is defined to be the subgraph of $\mathcal{G}(\mathcal{X})$ induced by the set of j-nodes for $j \in \{i, \ell - i\}$.

Definition 10. *Any representing $\mathcal{G}(\mathcal{X})$ of a circular code $\mathcal{X}$ is a set of $\lfloor \frac{\ell}{2} \rfloor$ disjoint graph components. Let us assume that $\mathcal{X} \subseteq \Sigma^{\ell}$ is a finite block code, and $\mathcal{G}(\mathcal{X})$ is the representing graph of $\mathcal{X}$. This definition denotes $\mathcal{C}_1(\mathcal{X}) \ldots \mathcal{C}_{\lfloor \frac{\ell}{2} \rfloor}(\mathcal{X})$ as the $\lfloor \frac{\ell}{2} \rfloor$ components of $\mathcal{G}(\mathcal{X})$ so that:*

$$\mathcal{G}(\mathcal{X}) = \bigcup_{i=1}^{i \leq \lfloor \frac{\ell}{2} \rfloor} \mathcal{C}_i(\mathcal{X})$$

The definition of a graph i-components $\mathcal{C}_i(\mathcal{X})$ for all $i \in \{1, \ldots, \lfloor \frac{\ell}{2} \rfloor\}$ associated with a code demands that every node in $\mathcal{C}_i(\mathcal{X})$ must be a j-node in $\mathcal{G}(\mathcal{X})$ with $j \in \{i, \ell - i\}$. A $\mathcal{C}_i(\mathcal{X}) = (V_i(\mathcal{X}), E_i(\mathcal{X}))$ is defined with a set of vertices $V_i(\mathcal{X}) \subseteq V(\mathcal{X})$ and a set of arcs $E_i(\mathcal{X}) \subseteq E(\mathcal{X})$. The following definition is similar to the previous definition of $\mathcal{G}$, but the sets of arcs and vertices are reduced to a fixed $i \in \{1, 2, ..., , \lfloor \frac{\ell}{2} \rfloor\}$:

- $V_i(\mathcal{X}) \coloneqq \{b_1 \ldots b_i, b_{i+1} \ldots b_{\ell} \, : \, b_1 b_2 b_3 \ldots b_{\ell} \in \mathcal{X}\}$
- $E_i(\mathcal{X}) \coloneqq \{b_1 \ldots b_i \rightarrow b_{i+1} \ldots b_{\ell} \, : \, b_1 b_2 \ldots b_{\ell} \in \mathcal{X}\}$

Example 10.1. *Let us assume that $\mathcal{X} = \{ACCA, CAGT\}$ is a circular tetranucleotide code.*

By definition, $\mathcal{X}$ is 4-letter code. Hence, $\mathcal{G}(\mathcal{X})$ can be separated into two disjoint graph components. These graph components are the $\mathcal{C}_1(\mathcal{X})$ and the $\mathcal{C}_2(\mathcal{X})$.

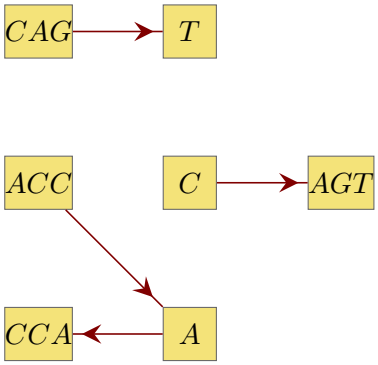


Figure 1.14: $\mathcal{C}_1(\mathcal{X})$ of the representing graph $\mathcal{G}(\mathcal{X})$ of the code $\mathcal{X} = \{ACCA, CAGT\}$

Figure 1.15: $\mathcal{C}_2(\mathcal{X})$ of the representing graph $\mathcal{G}(\mathcal{X})$ of the code $\mathcal{X} = \{ACCA, CAGT\}$

Figure 1.14 & 1.15: Show the two disjoint components $\mathcal{C}_1(\mathcal{X})$ and $\mathcal{C}_2(\mathcal{X})$ of $\mathcal{G}(\mathcal{X})$.

1.2.7 Symmetry group over the elements of the alphabet

In addition to the symmetry group S_ℓ presented above, we need to introduce another group S_Σ. This group acts on the elements in the alphabet Σ.

$$S_\Sigma \coloneqq \{\pi : \Sigma \to \Sigma : \pi \textit{ is bijective}\}$$

Similar to $(S_\ell, \circ)$, the group $(S_\Sigma, \circ)$ is endowed with the usual group operation of composition. Let us recall that $n \coloneqq |A|$. The set S_Σ has $n!$ elements and is the same for any word length $\ell \in \mathbb{N}$.

We now draw the attention to group $S_{\mathfrak{B}}$ acting on the elements of the genetic alphabet $\mathfrak{B}$, in particular, on a subgroup $\mathcal{L}$ of $S_{\mathfrak{B}}$ which was presented in the articles [19, 20]. The group $\mathcal{L}$ is isomorphic to a Dihedral group, and each element in $\mathcal{L}$ maintains the codon-anti-codon relationship and all circular related properties in the target. The group $(\mathcal{L}, \circ)$ is formally defined as:

$$\mathcal{L} \coloneqq \{\pi : \mathfrak{B} \to \mathfrak{B} : \pi \textit{ is bijective}\}$$

Let us list the mappings in $\mathcal{L}$. We will use the standard notation to define the mappings. For instance, the mapping $\pi_{ATCG} : (A, T, C, G) \to (T, C, G, A)$ maps $\pi_{ATCG}(A) = T$, $\pi_{ATCG}(T) = C$, $\pi_{ATCG}(C) = G$ and $\pi_{ATCG}(G) = A$. The first four mappings are named after their chemical characters. Note that they are invariant

in terms of the chemical properties of nucleotides [37].
Identity:

$$\text{I (or id)} : (A, T, C, G) \to (A, T, C, G);$$

Strong/Weak (SW) or complementary mapping:

$$\text{SW (or c)} : (A, T, C, G) \to (T, A, G, C);$$

Pyrimidine/Purine (YR0) mapping:

$$\text{YR (or p)} : (A, T, C, G) \to (G, C, T, A);$$

Keto/Amino (KM) mapping:

$$\text{KM (or r)} : (A, T, C, G) \to (C, G, A, T).$$

π_{CG} mapping:

$$\pi_{CG} : (A, T, C, G) \to (A, T, G, C).$$

π_{AT} mapping:

$$\pi_{AT} : (A, T, C, G) \to (T, A, C, G).$$

π_{ACTG} mapping:

$$\pi_{ACTG} : (A, T, C, G) \to (C, G, T, A).$$

π_{AGTC} mapping:

$$\pi_{AGTC} : (A, T, C, G) \to (G, C, A, T).$$

It has been observed that there are exactly 216 maximal self-complementary C^3 codes over the genetic alphabet [1, 58]. By the means of the group $\mathcal{L}$, these 216 codes can be classified into 27 equivalence classes [20] where each class represents an orbit of a code under the group. In anticipation of the following section 1.2.8, we would like to mention that there is one distinct code among the 216 codes in genetic sequences. This code is called the X-code [1] and will be described in more detail in section 1.2.8.

1.2.8 Block codes in DNA and RNA

While the theory that the entire modern genetic code is a comma-free code has been refuted, this does not prove that there are no comma-free codes in the genetic code at all. Some trinucleotide comma-free codes have been statistically traced in genes. This led to the hypothesis that comma-free codes may apply in a primitive translation machinery. The three leading theories are based on the three comma-free codes: *RRY, RNY* and *GNC* [45, 69, 11]. The first one is the *RRY* code:

$$\text{RRY} = \{AAC,\ AAT,\ AGC,\ AGT,\ GAC,\ GAT,\ GGC,\ GGT\}$$

which encodes for four amino acids: *Asn, Asp, Gly* and *Ser*. The *RRY* code is a subset of the *RNY* code:

$$\text{RNY} = \{AAC,\ AAT,\ AGC,\ AGT,\ GAC,\ GAT,\ GGC,\ GGT,\ ACC,\ ACT,\ ATC,\ ATT,\ GCC,\ GCT,\ GTC,\ GTT\}$$

which extends the four amino acids encoded by *RRY* to a set of eight amino acids: *Ala, Asn, Asp, Gly, Ile, Ser, Thr* and *Val*. The third comma-free code is the *GNC* code which is also a subset of the *RNY* code.

$$\text{GNC} = \{GAC,\ GGC,\ GCC,\ GTC\}$$

The *GNC* code encodes for the amino acids: *Ala, Asp, Gly* and *Val*

At this point, we would like to make a short excursion to the evolution theory. If we have a closer look at the amino acids encoded by *RRY, RNY* and *GNC* it can be seen that they almost fit to Higgs' suggested precursor amino acids *Gly, Ala, Asp, Glu* and *Val* [44]. Such a concordance of facts suggests that both the coevolution theory and the adaptive theory had an influence on the evolution of the genetic code.

A recently developed theory based on statistically elaborated evidence indicates the existence of a circular code in the genetic model. In 1996, Arquès and Michel were able to identify a set of 20 codons. This set falls under the definition of a code and is called X-code. The X-code was discovered by a statistical analysis of genes of bacteria, archaea, eukaryotes, plasmids and viruses. [1].

$$X = \{\text{AAC, AAT, ACC, ATC, ATT, CAG, CTC, CTG, GAA, GAC, GAG, GAT, GCC, GGC, GGT, GTA, GTC, GTT, TAC, TTC}\}$$

$$\alpha_1(X) = \{\text{ACA, ATA, CCA, TCA, TTA, AGC, TCC, TGC, AAG, ACG, AGG, ATG, CCG, GCG, GTG, TAG, TCG, TTG, ACT, TCT}\}$$

$$\alpha_2(X) = \{\text{CAA, TAA, CAC, CAT, TAT, GCA, CCT, GCT, AGA, CGA, GGA, TGA, CGC, CGG, TGG, AGT, CGT, TGT, CTA, CTT}\}$$

Concatenations of words of this X-code preferentially appear in genes in the 0-frame [70, 18], while $\alpha_1(X)$ preferentially appears in the +1 frame and $\alpha_2(X)$ in the +2 frame. A concatenation of words of one code without separation in one frame is called a motif (see Definition 11). The proportion of the motifs found in genes compared to non-coding regions was significantly higher [55]. Astonishingly, the X-code has proven to be a circular self-complementary C^3 code [1, 51, 70].

Definition 11. *Let x and ℓ be positive integers, $w = w_1 \cdots w_x \in \Sigma^+$ be a non-empty sequence over the alphabet Σ and $\mathcal{X} \subseteq \Sigma^\ell$ a ℓ-letter block code over the same alphabet. A motif is a concatenated set of words in w*

$$w_1 \cdots w_k \cdots w_{k+j} \cdots w_x \text{ where } k, j > 1 \in \mathbb{N}$$

so that $w_k, \ldots, w_{k+j} \in \mathcal{X}$ implies that $w_k \cdots w_{k+j}$ is a motif of $\mathcal{X}$ or a $\mathcal{X}$-motif.

1.3 Summary of Chapter 1

In this chapter the basic principles on which this work is based are summarized. Both, the mathematical and biological aspects focus on the synthesis of proteins in cells. There are two different cell types, the prokaryotic and the eukaryotic cells. The prokaryotic cells are found in organisms of the domains of archaea and bacteria whereas the eukaryotic cells in organisms of the domain of eukaryotes. In both cell types, hereditary information is stored in DNA. Simplified, DNA is a nucleotide based sequence. Each nucleotide stands for one of four bases: adenine (A), guanine (G), cytosine (C) or thymine (T). The information stored in DNA is used for the synthesis of proteins. For this purpose a duplicate of parts of the heritage information stored in DNA is created, called tRNA. However, it is not an exact copy. While DNA consists of two strands of nucleotides, tRNA only consists of a single strand of nucleotides. Additionally, in tRNA, the base uracil (U) is used instead of thymine (T). After the pre-translation process (only in eukaryotic cells) the tRNA is translated by the ribosome into a polypeptide chains (chain of amino acids). These chains fold into proteins which regulate and control the main purpose of the cell itself. The translation as well as the copying of the DNA is based on a system of complementary mapping denoted as function $c(\cdot)$. Where:

$$c(A) \coloneqq T \,(or\ U) \text{ and } c(C) \coloneqq G \text{ and vice versa.}$$

During the translation in the ribosome, three tRNA bases are considered to be one word. Such a word consisting of three bases is called codon and is translated via a translation table into one of the 20 corresponding amino acids or a stop signal. Most living organisms use the same translation table, the standard genetic code (SGC) (see figure 1.6). Numerous hypotheses claim that the SGC evolved from a primitive genetic code via the last common ancestor code (LUCA), making it the most robust against translation errors.

Such a hypothetical evolutionary approach would correspond with the use of a block code derived from code theory as an error correction mechanism in translation. Evidence of such a block code in form of the so called X-code has been found in coding sequences of different species. This X-code is one of 216 maximal circular self-complementary C^3 codes. Specifically, the X-code is maximal,

with a size of 20 codons, which cannot be contained in a circular code of higher cardinality. Furthermore, it is self-complementary, since for any codon in the X-code the responding anti codon is also in the X-code. Finally, it is C^3, *i.e.* all circular permutations of the X-code are also maximal circular codes[1]. A circular code is defined so that any concatenation of words of the code written on a cycle can only be decomposed into words of the code in one reading-frame. Another approach uses comma-free codes. These codes are more restrictive than circular codes and detect a frameshift error immediately. The three leading theories on comma-free codes in genes are based on the three comma-free codes: *RRY, RNY* and *GNC* [45, 69, 11].

Furthermore, new mathematical results of the investigation of circular codes are presented. These results provide new research tools from graph theory and group theory. Let us recall the in this thesis two most frequently used tools: Firstly, is a directed graph which can be associated to a circular code. Such a graph is defined as:

Definition (copy of Definition 9)**.** *Let $\ell \in \mathbb{N}$, and let $\mathcal{X} \subseteq \Sigma^\ell$ be an finite ℓ-letter block code. A $\mathcal{G}(\mathcal{X}) = (V(\mathcal{X}), E(\mathcal{X}))$ is defined with a set of vertices $V(\mathcal{X})$ and a set of arcs $E(\mathcal{X})$. Let us define the graph $\mathcal{G}$ associated with a code $\mathcal{X}$.*

- $V(\mathcal{X}) \coloneqq \{N_1 \dots N_i, N_{i+1} \dots N_\ell \,:\, N_1 \dots N_\ell \in \mathcal{X}$ *for every* $i \in \{1, ..., \lfloor \frac{\ell}{2} \rfloor\}\}$
- $E(\mathcal{X}) \coloneqq \{w_1 \to w_2 \,:\, w_1 \dot{w}_2 \in \mathcal{X}$ *and* $(w_1, w_2) \in V(\mathcal{X})\}$

The graph $\mathcal{G}(\mathcal{X})$ is the graph associated *with $\mathcal{X}$. For each $i \in \{1, \dots, \ell\}$, the vertices of $\mathcal{G}(\mathcal{X})$ that correspond with words of length i are referred to as* i-nodes*.*

Secondly, the circular equivalence classes. These classes are defined as an orbit under a group of all circular permutation.

Definition (copy of Definition 3)**.** *Let $(\mathcal{A}_\ell, \circ)$ be a group. Where $\mathcal{A}_\ell \coloneqq \{\alpha_0, \dots, \alpha_{\ell-1}\}$ and $\circ$ denotes the composition of mappings. Be $\alpha_0(\cdot)$ the identity element. Then, $\mathcal{A}_\ell$ is a group acting on Σ^ℓ:*

$$\alpha \colon \mathcal{A}_\ell \times \Sigma^\ell \to \Sigma^\ell$$

All words in Σ^ℓ can be separated into so-called circular permutation classes by the means of $\mathcal{A}_\ell$. We define a circular permutation class as an orbit of $\mathcal{A}_\ell$ acting on a word $w \in \Sigma^\ell$.

$$\mathcal{A}_\ell \cdot w = \{\alpha \cdot w \mid \alpha \in \mathcal{A}_\ell\}$$

The maximum size of an orbit, and therefore a circular permutation class, is equal to the cardinality of $\mathcal{A}_\ell$ which is ℓ. Hence, we donate a circular permutation class that has ℓ members as a complete circular permutation class. The other circular permutation classes are denoted as incomplete circular permutation classes.

Chapter 2

Properties of circular codes

The previous chapter illustrates the structure of circular codes (see section 1.2.3) as block codes and introduces the X-code (see section 1.2.8) which was the first verification of such codes in genetic sequences. The introduction presents clear evidence pointing to an important role of the X-code in protein synthesis. Since the X-code also belongs to the class of maximal self-complementary circular codes, the foremost objective of this chapter is the expansion of knowledge of the class of self-complementary circular codes and the class of maximal circular codes.

In the first section of this chapter 2.1, we investigate self-complementary codes. For this investigation, we will apply deep graph theory based on graphs associated with a code (see section 1.2.6). First, we link the minimum size of a code motif needed to ensure the correct reading-frame with the length and structure of the longest paths in the graphs associated with the self-complementary circular code. Subsequently, we formulate conditions to fully classify self-complementarity in graphs associated with a circular self-complementary code of size at least 18. Most results presented in this section have already been published in an article [24].

The second section, 2.2, examines the entire circular code family from a more general point of view. To investigate the full potential of circular codes and their role as hypothetical ancestor codes, this section focuses on circular codes over arbitrary finite alphabets and/or arbitrary word lengths.. As another relevant group of block codes, the comma-free codes are also examined more closely as a "selection factor that guides the search for

...

hypothetical ancestor codes"[59].

2.1 Self-complementary circular codes in coding theory

Reading the amino acids encoded in the mRNA in the normal reading-frame is essential for protein synthesis. Due to the start codons in the mRNA we can assume that the process starts the reading in the correct reading-frame. It is often proposed that there was or still is a mechanisms to maintain this reading-frame [1, 59]. These proposals often refer to two types of codes: The comma-free codes [11] and the circular codes. In this section we work on the circular codes. Such codes are capable of synchronizing, maintaining and retrieving the normal reading-frame. As mentioned in the introduction, the X-code has been identified as a representative of the circular codes in genes of bacteria, archaeae, eukaryotes, plasmids and viruses [1, 17]. It has not yet been possible to explain how the X-code is applied in the genome because it appears as individual motifs (see Definition 11) in the sequences. The motivation of this section is to get a deeper understanding of the motifs of the X-code.

Since the X-code is one of the 216 maximal self-complementary C^3 codes (see section 1.2.7), the X-code must have the identical characteristics as the codes of this subclassses of circular codes. It therefore led to a closer observation of the 216 codes. The data obtained guided the theorems and their corresponding proofs presented in this section. The focus lies on the coding theoretical properties of the self-complementary maximal codes $\mathcal{X} \subset \mathfrak{B}^3$ with a word length of $\ell = 3$ over a fixed alphabet $\mathfrak{B} = \{A, C, G, T\}$ of cardinality $n = 4$. These codes have been studied by representing them through directed graphs and then applying deep graph theory. The results in the subsequent sections will elaborate on the number of nucleotides in a sequence required for frameshift recognition, and the full classification of graphs associated with self-complementary code.

Most results in this section are published in the article "Self-complementary circular codes in coding theory"[24].

2.1.1 Frameshift robustness on self-complementary C^3 codes

Let $\mathcal{X} \subset \mathfrak{B}^3$ be a strong comma-free trinucleotide code. Since $\mathcal{X}$ is strong comma-free, all sequences $b_1 \dots b_i$ of words $b_1b_2b_3, \dots, b_{i-2}b_{i-1}b_i \in \mathcal{X}$ can be described as frameshift robust. Recalling the strong comma-free definition, this is because neither $b_2b_3b_4$ nor $b_3b_4b_5$ can be in $\mathcal{X}$. However, let us assume that $\mathcal{X}$ is a circular code and not strong comma-free. For each sequence $b_1 \dots b_i$ of concatenated words $b_1b_2b_3, \dots, b_{i-2}b_{i-1}b_i \in \mathcal{X}$ written on a circle, a frameshift error cannot occur undetected. Nonetheless, an undetected frameshift error can occur in the same linearly written sequence if the motif of a code is not long enough. The minimum length

needed for a motif to avoid undetected frameshift errors does not only depend on the construction of the code, but also on the concatenated words used for the motif. This section introduces a classification of codes based on the weakest link in a code, i.e. the number of nucleotides in a motif needed to detect a frameshift for any code motif.

Observation 2.1.1. *Let j be a positive integer and $\mathcal{X} \subset \mathfrak{B}^3$ be a trinucleotide circular code. Any sequence $w_s = w_1 w_2 \cdots w_j \in \mathcal{X}^j$ of words $w_1, w_2, \ldots, w_j \in \mathcal{X}$ written in a circle have only one circular $\mathcal{X}$-decomposition in the correct reading-frame. However, the same sequence can be decomposed into the same number of words of $\mathcal{X}$ in two frames if w_s is considered a motif contained in a longer sequence.*

To prove the Observation 2.1.1 we reconstruct the observed scenario in Construction method 2.1.1. It shows that the prefix, i.e. the nucleotides directly before a motif, and the suffix, i.e. the nucleotides directly after a motif, have an influence on the error detection mechanism of circular codes. In the constructed case used in the proof, the code motif in the correct reading-frame has the same length (number of nucleotides) as the motif in the +1 reading-frame.

Construction method 2.1.1. *Let m be a positive integer and $\mathcal{X} \subset \mathfrak{B}^3$ be a trinucleotide circular code and $w_s \coloneqq b_1 \cdots b_i \cdots b_{i+j} \cdots b_m \in \mathfrak{B}^m$ a sequence where $i \in \{(3n)+1 : n \in \mathbb{N}\}$, $j \in \{(3n)-1 : n \in \mathbb{N}^+\}$ and $i + j < m$. Let us further assume that the words*

$$b_i b_{i+1} b_{i+2}, \ldots, b_{i+j-2} b_{i+j-1} b_{i+j} \in \mathcal{X}$$

are a motif of $\mathcal{X}$ in the correct reading-frame. Without loss of generality, we assume $j = 5$, from which follows that $w_1 \coloneqq b_i b_{i+1} b_{i+2}$ and $w_2 \coloneqq b_{i+3} b_{i+4} b_{i+5}$, with reference to the construction $w_1, w_2 \in \mathcal{X}$. However, the codon

$$b_{i+1} b_{i+2} b_{i+3}, b_{i+4} b_{i+5} b_{i+6} \in \mathfrak{B}^3$$

can be in $\mathcal{X}$ without losing circularity. This case forces that

$$b_2 \ldots b_{i+1} b_{i+2} b_{i+3} b_{i+4} b_{i+5} b_{i+6} \ldots b_m$$

is a valid $\mathcal{X}$ motif in the +1 frame with the same length as the $\mathcal{X}$ motif in the normal frame.

This paragraph refers to the results of the paper "Self-complementary circular codes in coding theory" [24]. Considering the previous observation, it is evident that the prefix and suffix of a code motif are important factors for the reliability of the code motif. To specify the correct notation of a code motif in each frame, we define the $\mathcal{X}$-frame, d_p and d_s in Definition 12.

Definition 12. *Let $\mathcal{X} \subset \mathfrak{B}^3$ be a trinucleotide code and $x \geq 3$ be an integer so that $b_1 \dots b_x \in \mathfrak{B}^x$ a sequence of nucleotides. The sequence $b_1 \dots b_x$ is constructed so that it can be divided into 3 elements:*

$$b_1 \dots b_x = d_p c_1 \dots c_l d_s$$

where $x \geq 3$, $l = x - (|d_p| + |d_s|)$, $c_i c_{i+1} c_{i+2} \in \mathcal{X}$ for $i = 1, 4, 7, ..., l-2$ is a $\mathcal{X}$ motif, $d_p \in \{\epsilon, b_1, b_1 b_2\}$ and $d_e \in \{\epsilon, b_{x-1} b_x, b_x\}$, ϵ being the empty word.

According to Definition 12, the prefix d_p and the suffix d_s of the sequence do not need to be a prefix or suffix of a codon in $\mathcal{X}$. Only the central part, denoted as $c_1 \dots c_l$, consists of words from $c_i c_{i+1} c_{i+2} \in \mathcal{X}$ for $i = 1, 4, 7, ..., l-2$. As Fimmel et al. state, "[t]his approach contrasts the notion of unambiguous words defined in [50] and makes the notion of $\mathcal{X}$-frame and later on reading-frame applicable to arbitrary sequences, i.e., not entirely consisting of trinucleotides from $\mathcal{X}$." [24]

With this definition of $\mathcal{X}$-frames, it is possible to associate the weak point like the one depicted in Observation 2.1.1, with a path in the associated $\mathcal{G}(\mathcal{X})$ graph. These path-related graph properties are defined in Definition 13.

Definition 13. *Let $\mathcal{X} \subset \mathfrak{B}^3$ be a trinucleotide circular code and $\mathcal{G}(\mathcal{X})$ be its associated graph. Let $p : v_1 \to \dots \to v_n$ be a path in $\mathcal{G}(\mathcal{X})$, where each v_i is either a 1-node of a 2-node for $i = 1. \dots, n$. Then the arrow length $l_a(p)$ is defined as $n-1$. Furthermore, with $l_{max}(\mathcal{X})$ we define the maximum arrow length of a path, i.e. the length of a longest path, in the associated graphic $\mathcal{G}(\mathcal{X})$.*

- *the arrow-length $l_a(p)$ is defined as $l_a(p) = n - 1$;*
- *the length of a longest path, in the associated graph $\mathcal{G}(\mathcal{X})$ is defined as $l_{max}(\mathcal{X})$.*
- *the word associated with p is defined as $w(p) = v_1 \dots v_n$, the concatenation of the labels of p;*
- *the word length $l_w(p)$ is defined as $|w(p)|$, the length of the word associated with p.*

While most $\mathcal{X}$-frames cannot be decomposed into words of the same code in more than one reading-frame, there are rare scenarios where this happens, leading to overlaps. Such a case of overlapping $\mathcal{X}$-frames is the weak point of the circular codes described in Observation 2.1.1.

Observation 2.1.2. *Let $\mathcal{X} \subset \mathfrak{B}^3$ be a trinucleotide circular code and and $x \geq 4$ be an integer so that $b_1 \cdots b_x \in \mathfrak{B}^x$ a sequence of nucleotides. If we assume that $b_1 \cdots b_x$ has 2 different possible $\mathcal{X}$-frames*

$$b_1 \cdots b_x = d_p c_1 \cdots c_l d_s = d'_p c'_1 \cdots c'_m d'_s \text{ so that } d_p \neq d'_p$$

with $c_i, c'_i \in \mathcal{X}$ *and* $d_p, d_s, d'_p, d'_s \in \{\epsilon\} \cup \mathfrak{B}^2 \cup \mathfrak{B}$*, then, there exists a path in* $\mathcal{G}(\mathcal{X})$ *associated with the overlapping sequences* $c_1 \cdots c_l$ *and* $c'_1 \cdots c'_m$*. The word associated with this path (see Definition 13 above) covers exactly the smallest subsequence of* $b_1 \cdots b_n$ *that contains both* $c_1 \cdots c_l$ *and* $c'_1 \cdots c'_m$*.*

To avoid overlapping $\mathcal{X}$-frames as in Observation 2.1.2, the sequences must be constructed in such a way that they are not associated with a path in $\mathcal{G}(\mathcal{X})$. Once the concatenation of words in a code is equal to the concatenation of labels of a path associated with the code, the $\mathcal{X}$-frame overlaps. This can generally be avoided by defining an upper bound.

Definition 14. *Let* $\mathcal{X} \subset \mathfrak{B}^3$ *be a trinucleotide circular code. We define the reading-frame number* $n_{\mathcal{X}}$ *of* $\mathcal{X}$ *as*

$$n_{\mathcal{X}} := min\{n \in \mathbb{N} | \text{ for all sequences of nucleotides } b_1 \dots b_n \text{ there is at most one possible } \mathcal{X}\text{-frame for } b_1 \dots b_n\}$$

This upper bound can be derived from the longest path of a graph. Suppose $\mathcal{X} \subset \mathfrak{B}^3$ is a circular code. Then, the *reading-frame number* $n_{\mathcal{X}}$ is considered valid, if:

$$n_{\mathcal{X}} \leq 2 \cdot l_{max}(\mathcal{X}) + 3$$

The following Theorem 2.1.1 proves this dependence. The proof of the theorem can be found in article [24].

Theorem 2.1.1. *(Theorem 5.9, [24]) Let* $\mathcal{X} \subset \mathfrak{B}^3$ *be a trinucleotide circular code and* $\mathcal{G}(\mathcal{X})$ *its associated graph. Then, the reading-frame number* $n_{\mathcal{X}}$ *satisfies* $n_{\mathcal{X}} \leq 2 \cdot l_{max}(\mathcal{X}) + 3$*.*

Applied theory of the reading-frame number

The definition of the *reading-frame number* allows for the classification of trinucleotide circular codes with respect to the associated $n_{\mathcal{X}}$. Let us recall that $n_{\mathcal{X}} \leq 2 \cdot l_{max}(\mathcal{X}) + 3$ which demonstrates that the *reading-frame number* depends on the longest path of a code $l_{max}(\mathcal{X})$.

Let $\mathcal{G}(\mathcal{X})$ be a graph associated with the circular code $\mathcal{X}$. Recalling theorem 1.2.1, $\mathcal{G}(\mathcal{X})$ must be acyclic. However, the maximum path length is not limited. Exceptions are comma-free codes (Theorem 1.2.2), where $l_{max}(\mathcal{X}) = 2$, and strong comma-free codes (Theorem 1.2.3), where $l_{max}(\mathcal{X}) = 1$.

	Code size $\|X\|$									
l_{max}	2	4	6	8	10	12	14	16	18	20
1	12	8	0	0	0	0	0	0	0	0
2	16	202	556	642	396	152	36	4	0	0
3	0	16	152	336	280	80	0	0	0	0
4	0	108	1344	5808	12048	14032	9800	4116	964	96
5	0	0	0	0	0	0	0	0	0	0
6	0	0	68	684	2352	3896	3568	1872	532	64
7	0	0	0	0	0	0	0	0	0	0
8	0	0	56	824	4024	9104	10920	7248	2536	368
Total	28	334	2176	8294	19100	27264	24324	13240	4032	528

Table 2.1: Growth function of self-complementary circular codes $\mathcal{X}$ of even cardinality $a = 2, 4, \ldots, 20$ as a function of the longest path length $l_{max}(\mathcal{X}) = 1, ..., 8$ in their associated graph $\mathcal{G}(\mathcal{X})$

Table 2.1 illustrates the number of self-complementary circular codes $\mathcal{X}$ of different size $|\mathcal{X}|$ depending on the length of a longest path $l_{max}(\mathcal{X})$. The brute force results show that $l_{max}(\mathcal{X}) \leq 8$. The following Theorem 2.1.2 explains this issue and fully characterises the possible values of $l_{max}(\mathcal{X})$ for non-maximum and maximum self-complementary circular codes.

Theorem 2.1.2. *(Theorem 4.2, [24]) Let $\mathcal{X} \subset \mathfrak{B}^3$ be a trinucleotide circular code. The following statements about the maximal path length $l_{max}(\mathcal{X})$ of a path are true:*

1. $1 \leq l_{max}(\mathcal{X}) \leq 8$;
2. *If $\mathcal{X}$ is self-complementary, then $l_{max}(\mathcal{X}) \in \{1, 2, 3, 4, 6, 8\}$, i.e., $l_{max}(\mathcal{X}) = 5, 7$ are excluded;*
3. *If $\mathcal{X}$ is maximal and self-complementary, then $l_{max}(\mathcal{X}) \in \{4, 6, 8\}$, i.e., in addition to (2), $l_{max}(\mathcal{X}) = 1, 2, 3$ are impossible.*

Proof. The proof is structured into the three statements from Theorem 2.1.2.

Claim (1) It is immediate since for $l_{max}(\mathcal{X}) \geq 9$ in a graph $\mathcal{G}(\mathcal{X})$ associated with a circular code, there is a path containing at least five vertices labeled by nucleotides. Note that with the construction of $\mathcal{G}(\mathcal{X})$, the labels of the vertices alternate between nucleotides and dinucleotides. Nonetheless, there are only four different bases in the alphabet $\mathfrak{B}$, hence two of the vertices must have the same label which yields a cycle in $\mathcal{G}(\mathcal{X})$, in contradiction to circularity. Thus, $1 \leq l_{max}(\mathcal{X}) \leq 8$.

Claim (2) Let $\mathcal{X}$ be a self-complementary circular code.

(i) Assume that $l_{max}(\mathcal{X}) \geq 4$ is odd. By construction of $\mathcal{G}(\mathcal{X})$, any path in $\mathcal{G}(\mathcal{X})$ starts with either a nucleotide or a dinucleotide. Moreover, the vertices of the path alternate between nucleotides and dinucleotides. Thus, if $l_{max}(\mathcal{X})$ is odd, then the longest path in $\mathcal{G}(\mathcal{X})$ must either be of the form

$$(I) \quad b_1 \to a_1 \to b_2 \to a_2 \to \cdots \to d_{n-1} \to l_n \to d_n$$

starting with a nucleotide b_1 and ending with a dinucleotide d_n, or

$$(II) \quad a_1 \to b_1 \to a_2 \to b_2 \to \cdots \to l_{n-1} \to d_n \to l_n$$

starting with a dinucleotide a_1 and ending with a nucleotide l_n. In fact, the following argument shows that both cases hold. Assume, without loss of generality, that the longest path is of the first form (I). By self-complementarity, we then obtain a *complementary and reversed* path

$$(III) \quad \overleftarrow{c(d_n)} \to c(l_n) \to \overleftarrow{c(d_{n-1})} \to \cdots \to \overleftarrow{c(a_2)} \to c(b_2) \to \overleftarrow{c(a_1)} \to c(b_1).$$

Since in (I) we have assumed that $l_{max}(\mathcal{X}) \geq 4$, at least three nucleotides, $b_1, b_2, b_3, \cdots$, appear. By circularity of the code $\mathcal{X}$, all these nucleotides have to be different. Otherwise, the path would contain a cycle. Similarly, path (III) has at least three different nucleotides $c(l_n), c(l_{n-1}), c(l_{n-2}), \cdots$. However, there are only four nucleotides in the alphabet $\mathfrak{B}$, hence, there must be $i, j \leq n$ such that $l_i = c(l_j)$. Seeing that path (I) starts with a nucleotide and path (III) starts with a dinucleotide, the two paths

$$(I') \qquad b_1 \to a_1 \to b_2 \to a_2 \to \cdots \to d_{i-1} \to l_i$$

$$(III') \quad \overleftarrow{c(d_n)} \to c(l_n) \to \overleftarrow{c(d_{n-1})} \to \cdots \to \overleftarrow{c(d_j)} \to c(l_j)$$

must have different lengths. Without loss of generality, assume that (III') is the longer path. Yet, then the path

$$\overleftarrow{c(d_n)} \to c(l_n) \to \overleftarrow{c(d_{n-1})} \to \cdots \to \overleftarrow{c(d_j)} \to c(l_j) = l_i \to d_i \to \cdots \to d_{n-1} \to l_n \to d_n$$

has a length greater than $l_{max}(\mathcal{X})$ - which is a contradiction.

(ii) The following Examples 14.1, 14.2 and 14.3 show that $l_{max}(\mathcal{X}) = 1, 2, 3$ exist for self-complementary circular codes that are not maximal.

Claim (3) Let $\mathcal{X}$ be a maximal self-complementary circular code.

(i) If $l_{max}(\mathcal{X}) \leq 2$ is true, then $\mathcal{X}$ is comma-free. However, there are no self-complementary comma-free codes of size 20 (Table 7 in Michel et al., 2008). Thus, according to the claim (2), $l_{max}(\mathcal{X}) \in \{3, 4, 6, 8\}$.

(ii) Assume now that $l_{max}(\mathcal{X}) = 3$. By maximality and circularity, $\mathcal{X}$ must contain exactly one element in each equivalence class $\{b_1b_2b_3,\ b_2b_3b_1,\ b_3b_1b_2\}$ for every trinucleotide $b_1b_2b_3$. Hence, $\mathcal{X}$ must contain one trinucleotide from $\{AAT,\ ATA,\ TAA\}$ and one complementary trinucleotide from $\{ATT,\ TTA,\ TAT\}$. It is apparent that each combination yields a path of the form $A \to a_1 \to T$ or $T \to a_1 \to A$ for some dinucleotide a_1. Similarly, we get a path of the form $C \to a_2 \to G$ or $G \to a_2 \to C$ for some dinucleotide a_2. Without loss of generality, assume that $A \to a_1 \to T$ and $C \to a_2 \to G$ are paths in $\mathcal{G}(\mathcal{X})$. Clearly, the four trinucleotides Aa_1, a_1T, Ca_2, a_2G are all different. Hence, $\mathcal{X}' = \mathcal{X}\backslash\{Aa_1, a_1T, Ca_2, a_2G\}$ has 16 elements. Assume that there is a trinucleotide $dC \in \mathcal{X}'$, d being a dinucleotide, then also $Gc(d) \in \mathcal{X}'$ by self-complementarity. Consequently, we get a path $d \to C \to a_2 \to G \to c(d)$ of length four - which is, again, a contradiction.

Similarly, we cannot have trinucleotides of the form $dA, Td, Gd \in \mathcal{X}'$. So, no trinucleotide in $\mathcal{X}'$ starts with T or G and no trinucleotide ends with C or A. Hence, $\mathcal{X}' \subseteq S = \{N_1N_2N_3 \mid N_2 \in \mathfrak{B}, N_1 \in \{A, C\}, N_3 \in \{G, T\}\}$. Clearly, $\mid S \mid = 16$. However, the four trinucleotides Aa_1, a_1T, Ca_2, a_2G are also in S, but excluded from $\mathcal{X}'$, so $\mid \mathcal{X}' \mid \leq 12$ - a contradiction.

□

Example 14.1. *The code $\mathcal{X}_1 = \{TGA,\ TCA\}$ of size two is a self-complementary circular code with the longest path length $l_{max}(\mathcal{X}_1) = 1$, e.g. $TC \to A$, $T \to GA$, etc. (Figure 2.1).*

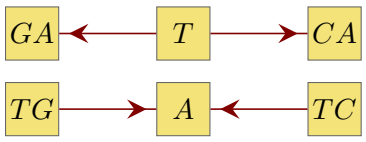


Figure 2.1: Graph $\mathcal{G}(\mathcal{X}_1)$ of the self-complementary circular code $\mathcal{X}_1 = \{TGA,\ TCA\}$ of size two with the longest path length $l_{max}(X_1) = 1$.

Example 14.2. *The code $\mathcal{X}_2 = \{TGC,\ GCA,\ CTC,\ GAG\}$ of size four is a self-complementary circular (even comma-free) code with the longest path length $l_{max}(\mathcal{X}_2) = 2$, e.g. $T \to GC \to A$, $GA \to G \to AG$, $GA \to G \to CA$, etc. (Figure 2.2).*

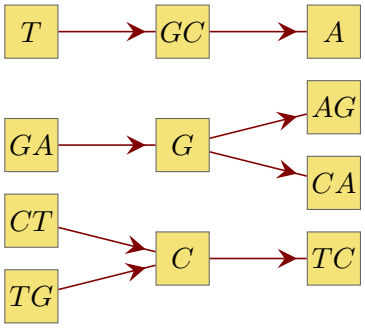


Figure 2.2: Graph $\mathcal{G}(\mathcal{X}_2)$ of the self-complementary circular code $\mathcal{X}_2 = \{TGC,\ GCA,\ CTC,\ GAG\}$ of size four with the longest path length $l_{max}(\mathcal{X}_2) = 2$.

Example 14.3. *The code $\mathcal{X}_3 = \{TTG,\ TGG,\ GTC,\ GAC,\ CCA,\ CAA\}$ of size six is a self-complementary circular code with the longest path length $l_{max}(\mathcal{X}_3) = 3$, e.g. $T \to TG \to G \to TC$, $GT \to C \to CA \to A$, etc. (Figure 2.3).*

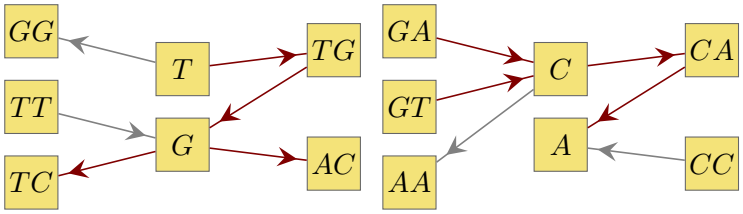


Figure 2.3: Graph $\mathcal{G}(\mathcal{X}_3)$ of the self-complementary circular code $\mathcal{X}_3 = \{TTG,\ TGG,\ GTC,\ GAC,\ CCA,\ CAA\}$ of size six with the longest path length $l_{max}(\mathcal{X}_3) = 3$.

As established in Theorem 2.1.2, if $\mathcal{X} \subset \mathfrak{B}^3$ is a maximal self-complementary circular code, then $l_{max}(\mathcal{X}) \in \{4, 6, 8\}$. In Theorem 2.1.3, this characterisation is refined for maximal self-complementary circular codes.

Theorem 2.1.3. *(Theorem 4.7, [24]) Let $\mathcal{X} \subset \mathfrak{B}^3$ be a maximal self-complementary trinucleotide circular code. Then the following statements hold:*

1. *If $l_{max}(\mathcal{X}) = 4$, then the longest paths are of the form*

$$a_1 \to b_1 \to a_2 \to b_2 \to a_3$$

2. *If $l_{max}(\mathcal{X}) = 6$, then the longest paths are of the form*

$$b_1 \to a_1 \to b_2 \to a_2 \to b_3 \to a_3 \to b_4$$

3. If $l_{max}(\mathcal{X}) = 8$, then the longest paths are of the form

$$a_1 \to b_1 \to a_2 \to \cdots \to a_4 \to b_4 \to a_5$$

where $b_i \in \mathfrak{B}$ are 1-nodes and $a_i \in \mathfrak{B}^2$ are 2-nodes for any $i \in \mathbb{N}$.

Proof. Claim (1): Let $l_{max}(\mathcal{X}) = 4$ and assume that $b_1 \to a_1 \to b_2 \to a_2 \to b_3$ is the longest path in $\mathcal{G}(\mathcal{X})$. Since the path is maximal, there is no trinucleotide of the form db_1 and no trinucleotide of the form b_3d in $\mathcal{X}$. Each self-complementary maximal circular code must contain the codons $b_1c(b_1b_1)$ and $b_1b_1c(b_1)$.[1] It follows that $c(b_3) = b_1$ and $a_1, a_2 \in \{b_2, c(b_2)\}^2$. Note that all the nucleotides b_1, b_2, b_3 must be different by circularity. Thus, we have four possibilities for a_1, a_2, namely b_2b_2, $b_2c(b_2)$, $c(b_2)b_2$ and $c(b_2)c(b_2)$. As $b_2b_2b_2 \notin \mathcal{X}$ by circularity, we have the following options for the two trinucleotides: $a_1b_2 \in \mathcal{X}$ and $b_2a_2 \in \mathcal{X}$.

$$\begin{aligned} a_1b_2: &\quad b_2c(b_2)b_2 \quad c(b_2)b_2b_2 \quad c(b_2)c(b_2)b_2; \\ b_2a_2: &\quad b_2b_2c(b_2) \quad b_2c(b_2)b_2 \quad b_2c(b_2)c(b_2). \end{aligned}$$

If a_1b_2 or b_2a_2 equal $b_2c(b_2)b_2$, then self-complementarity yields $c(b_2)b_2c(b_2) \in \mathcal{X}$ and the word $c(b_2)b_2c(b_2)b_2c(b_2)b_2$ contradicts circularity. Excluding the combinations $c(b_2)b_2b_2$, $b_2b_2c(b_2)$ and $c(b_2)c(b_2)b_2$, $b_2c(b_2)c(b_2)$, because the trinucleotides are obviously circular permutations of each other, only two combinations remain: $c(b_2)b_2b_2$, $b_2c(b_2)c(b_2)$ and $c(b_2)c(b_2)b_2$, $b_2b_2c(b_2)$. Yet, in this case, self-complementarity also yields a contradiction to circularity, since e.g. the complementary trinucleotide of $c(b_2)c(b_2)b_2$ is in the same equivalence class as $b_2b_2c(b_2)$.

Claim (2): Let $l_{max}(\mathcal{X}) = 6$ and assume that $a_1 \to b_1 \to a_2 \to b_2 \to a_3 \to b_3 \to a_4$ is the longest path in $\mathcal{G}(\mathcal{X})$. By self-complementarity, there is the reversed complemented path

$$\overleftarrow{c(a_4)} \to c(b_3) \to \overleftarrow{c(a_3)} \to c(b_2) \to \overleftarrow{c(a_2)} \to c(b_1) \to \overleftarrow{c(a_1)}.$$

Here, the central nucleotides b_2 and $c(b_2)$ of the two paths are either the pair A and T or C and G. Therefore, it suffices to show that there are paths $A \to d \to T$ or $T \to d \to A$ and $C \to d \to G$ or $G \to d \to C$ in $\mathcal{G}(\mathcal{X})$, since then we will obtain a path of length eight combining the two paths, e.g.

$$a_1 \to b_1 \to a_2 \to b_2 \to d \to c(b_2) \to \overleftarrow{c(a_2)} \to c(b_1) \to \overleftarrow{c(a_1)}$$

contradicting $l_{max}(\mathcal{X}) = 6$. However, by maximality, the code $\mathcal{X}$ must contain exactly one trinucleotide of the class $\{ATT, TTA, TAT\}$ and its complementary trinucleotide, as well as exactly one trinucleotide from the class $\{GCC, CCG, CGC\}$

[1] The combination $c(b_1)b_1c(b_1)$ and $b_1c(b_1)b_1$ is not circular and a maximal circular code must contain one codon of each circular permutation class.

and its complementary trinucleotide. It is easily verifiable that in each case we obtain either a path of the form $A \to d \to T$ or $T \to d \to A$ and $C \to d \to G$ or $G \to d \to C$. For example, if $ATT \in \mathcal{X}$, then also $AAT \in \mathcal{X}$ and we get the path $A \to AT \to T$ in $\mathcal{G}(\mathcal{X})$.

Claim (3): Let $l_{max}(\mathcal{X}) = 8$ and assume that $b_1 \to a_1 \to b_2 \to a_2 \to b_3 \to a_3 \to b_4 \to a_4 \to b_5$ is the longest path in $\mathcal{G}(\mathcal{X})$. Then, evidently, two out of the five nucleotides b_1, b_2, b_3, b_4, b_5 must be equal, which yields a cycle in $\mathcal{G}(\mathcal{X})$, and contradicting the circularity of $\mathcal{X}$. □

With Theorem 2.1.3 the classification of the different path patterns is completed. An application of these patterns is proposed in Theorem 2.1.4.

Theorem 2.1.4. *(Theorem 5.11, [24]) Let $\mathcal{X} \subset \mathfrak{B}^3$ be a maximal self-complementary trinucleotide circular code and $\mathcal{G}(\mathcal{X})$ its associated graph. Let $p = p_{max}(\mathcal{X})$ be a path of maximal arrow-length (and, hence, word-length) in $\mathcal{G}(\mathcal{X})$ and let $l_w(p)$ be its word-length. Then, the following statements about the reading-frame number $n_{\mathcal{X}}$ are true:*

1. $n_{\mathcal{X}} = l_w(p) + 2$, *if* $p = a_1 \to b_1 \to \cdots \to b_k$ *or* $p = b_1 \to a_1 \to \cdots \to a_k$;
2. $n_{\mathcal{X}} = l_w(p) + 1$, *if* $p = a_1 \to b_1 \to \cdots \to a_k$;
3. $n_{\mathcal{X}} = l_w(p) + 3$, *if* $p = b_1 \to a_1 \to \cdots \to b_k$,

where b_i are 1-nodes and a_i are 2-nodes for any i.

Proof. See Appendix proof II.1. □

2.1.2 Consequences of section 2.1.1

Theorem 2.1.3 and Theorem 2.1.4 offer the possibility to classify trinucleotide maximal self-complementary circular codes by their longest path lengths. If we add the Theorem 2.1.2 we can conclude that there must be three equivalence classes. These classes are called X_4, X_6 and X_8. Where the index $i \in \{4, 6, 8\}$ of the class name X_i indicates the length of the longest path. The obtained results demonstrate that a X_4 code has a reading-frame number of $n_{\mathcal{X}} = 9$, a X_6 code has a reading-frame number of $n_{\mathcal{X}} = 12$ and a X_8 code has a reading-frame number of $n_{\mathcal{X}} = 15$

Non-self-complementary circular codes of any size less than 21 can be classified in the classes $X_1, X_2, X_3, X_4, X_5, X_6, X_7$ and X_8. Yet, not for all of these classes the reading-frame number can uniquly be predicted. The classes X_1, X_3, X_5, X_7 and X_8 have a unambiguous reading-frame number. See Theorem 2.1.3 Claim 3, a X_8 has always a reading-frame number of $n_{\mathcal{X}} = 15$. The odd classes X_1 to X_7

have also a unambiguous reading-frame number (see Theorem 2.1.4 Claim 2). The reading-frame numbers are $n_{\mathcal{X}} = 5$ for X_1, $n_{\mathcal{X}} = 8$ for X_3, $n_{\mathcal{X}} = 11$ for X_5 and $n_{\mathcal{X}} = 14$ for X_7. The codes in the classes X_2, X_4 and X_6 have ambiguous reading-frame numbers (see Theorem 2.1.4 Claim 1 & 3). The reading-frame numbers are $n_{\mathcal{X}} \in \{6, 7\}$ for X_2, $n_{\mathcal{X}} \in \{9, 10\}$ for X_4 and $n_{\mathcal{X}} \in \{12, 13\}$ for X_6.

This means that a X_2 code is comma-free and a X_1 code is even strong comma-free (see Theorem 1.2.2 and Theorem 1.2.3). In the final Chapter 6 an evolution hypothesis is presented in section 6.1, which combines the results of this section 2.1.1 and the results presented in the following Chapters 3-5. Another possible use case can be found in the section "Application: Reading frame of the maximal C^3 self-complementary circular code $\mathcal{X}$ identified in genes" of the article "Self-complementary circular codes in coding theory"[2] [24].

2.1.3 Self-complementarity as a graph property

The two strands of DNA are bounded by hydrogen bonds between complementary bases. Hence, the nucleotide code on the one strand is the reversed complement of the nucleotide code on the other strand. A block code that can be applied to both DNA strands is therefore a self-complementary code. While self-complementarity is an important biological property, a graph $\mathcal{G}(\mathcal{X})$ associated with a self-complementary code $\mathcal{X}$ is not able to represent this property as simply as circularity. Still, we were able to fully classify all graphs associated with a self-complementary code of a size of at least 18 with the use of recognizable conditions. Proposition 2.1.2 introduces the graph conditions that can indicate self-complementarity. After outlining this property, counterexamples of circular codes of size $|\mathcal{X}| \leq 17$ are constructed. The graphs associated with these codes satisfy the conditions, but the codes are non-self-complementary. On the contrary, the concluding Theorem 2.1.5 proves that the property must apply to all self-complementary circular codes of size $|\mathcal{X}| \geq 18$. All results except for Theorem 2.1.5 have been published in [24]. However, the Theorem 2.1.5 is unpublished.

Proposition 2.1.2. *Let $\mathcal{X} \subseteq \mathfrak{B}^3$ be a self-complementary trinucleotide code and $\mathcal{G}(\mathcal{X}) = (V(\mathcal{X}), E(\mathcal{X}))$ the graph associated to $\mathcal{X}$. Then*

1. *$V(\mathcal{X}) = \overleftarrow{c(V(\mathcal{X}))}$, i.e. for each nucleotide $l \in V(\mathcal{X})$ its complementary nucleotide $c(l) \in V(\mathcal{X})$ and for each dinucleotide $d \in V(\mathcal{X})$ its complementary reversed dinucleotide $\overleftarrow{c(d)} \in V(\mathcal{X})$;*

[2] As co-author of the article [24], I had no influence on the chapter "Application: Reading frame of the maximal C^3 self-complementary circular code $\mathcal{X}$ identified in genes" and therefore decided not to include it in this work.

2. $d^+(v) = d^-(\overleftarrow{c(v)})$ *for any vertex* $v \in V(\mathcal{X})$.

Where $d^+(\cdot)$ *is denoted as the in-degree and* $d^-(\cdot)$ *is denoted as the out-degree of a vertex in* $V(\mathcal{X})$.

Proof. Condition (1): Let $b_1b_2b_3 \in \mathcal{X}$. Since $\mathcal{X}$ is self-complementary, we have $c(b_3)c(b_2)c(b_1) \in \mathcal{X}$. Thus, according to the definition of $\mathcal{G}(\mathcal{X})$, $b_1, b_3, c(b_1), c(b_3) \in V(\mathcal{X})$ and $b_1b_2, b_2b_3, c(b_3)c(b_2), c(b_2)c(b_1) \in V(\mathcal{X})$. Hence, Condition (1) holds.

Condition (2): $[b_1, b_2b_3]$, $[b_1b_2, b_3] \in E(\mathcal{X})$ is equivalent to $[c(b_3)c(b_2), c(b_1)]$, $[c(b_3), c(b_2)c(b_1)] \in E(\mathcal{X})$. Hence, Condition (2) holds. □

Below, we construct a code that is non-self-complementary, but the associated graph $\mathcal{G}(\mathcal{X})$ satisfies the conditions (1) and (2) of Proposition 2.1.2.

Construction method 2.1.3. *Start with a trinucleotide* $b_1b_2b_3$ *and then choose a subsequent trinucleotide that starts with the complementary of the dinucleotide* b_2b_3*, but does not end with the complement of* b_1*. Continue this process until you choose a word which ends with the complementary of the dinucleotide* b_1b_2*, but does not start with the complement of* b_3*. While the code constructed this way will satisfy the two conditions* (1) *and* (2) *of Proposition 2.1.2, it is non-self-complementary.*

The following is a basic example of Construction Method 2.1.3.

Example 14.4. *The code* $\mathcal{X} = \{CAC, GAG, CTG, GTC\}$ *is non-self-complementary since, for example, it does not contain the complementary trinucleotide GTG of CAC. Yet, it is apparent that its associated graph satisfies the two conditions* (1) *and* (2) *from Proposition 2.1.2. The code* $\mathcal{X}$ *is comma-free and has been constructed using the preceding construction Method 2.1.3:* $CAC \rightsquigarrow GTC \rightsquigarrow GAG \rightsquigarrow CTG$.

However, Method 2.1.3 does not yield non-self-complementary codes of a size larger than eight, such that the associated graphs satisfy the two conditions (1) and (2) of Proposition 2.1.2.

Construction method 2.1.4. *To construct non-self-complementary codes that are larger than size eight and satisfy the two conditions* (1), (2) *of Proposition 2.1.2, the codes constructed by Method 2.1.3 can be combined.*

Using Method 2.1.4, Example 15.1 below shows that there are in fact codes of size 20, such that their associated graphs satisfy the two conditions (1) and (2) of Proposition 2.1.2, which are non-self-complementary, even strong non-self-complementary, and non-circular.

Definition 15. *A code* $\mathcal{X}$ *is strong non-self-complementary if for any trinucleotide* $w \in \mathcal{X}$*, the complementary trinucleotide* $\overleftarrow{c(w)} \notin \mathcal{X}$.

Example 15.1. *The code* $\mathcal{X}$ *of size* 20

$$\mathcal{X} = \{AAT,\ ACA,\ AGT,\ ATC,\ CAC,\ CCG,\ CGA,\ CTG,\ GAA,\ GAG,\ GCA,\ GGC,\ GTC,\ GTT,\ TAC,\ TCC,\ TCT,\ TGA,\ TGG,\ TTA\}$$

is strong non-self-complementary and non-circular, but its graph $\mathcal{G}(\mathcal{X})$ *satisfies the two conditions* (1) *and* (2) *of Proposition 2.1.2. Figure 2.4 displays the graph* $\mathcal{G}(\mathcal{X})$ *associated with* $\mathcal{X}$.

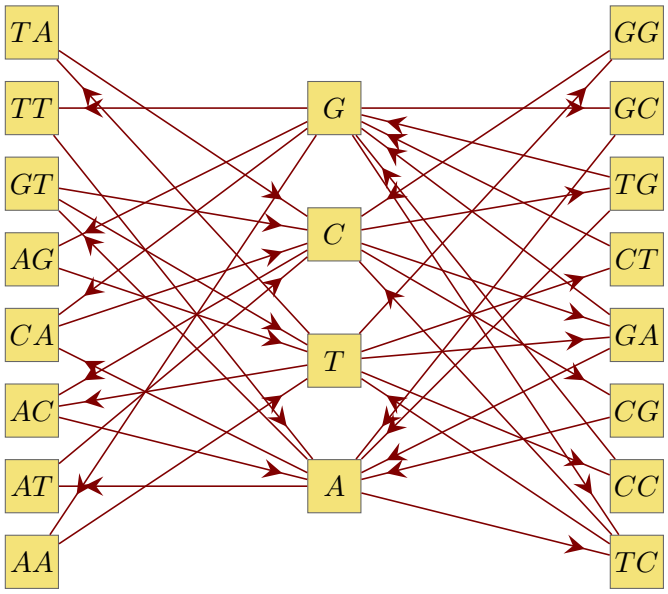


Figure 2.4: Graph $\mathcal{G}(\mathcal{X})$ of the strong non-self-complementary and non-circular code $\mathcal{X} = \{AAT,\ ACA,\ AGT,\ ATC,\ CAC,\ CCG,\ CGA,\ CTG,\ GAA,\ GAG,\ GCA,\ GGC,\ GTC,\ GTT,\ TAC,\ TCC,\ TCT,\ TGA,\ TGG,\ TTA\}$ of size 20 satisfying the two conditions (1) and (2) of Proposition 2.1.2.

The following Lemma 2.1.1 demonstrates how non-self-complementary circular codes of a size ≤ 16 can be constructed effortlessly with the help of an associated graph $\mathcal{G}(\mathcal{X})$ that satisfies conditions (1) and (2) of Proposition 2.1.2.

Lemma 2.1.1. *Let* $\mathcal{X}_1, \mathcal{X}_2 \subseteq \mathfrak{B}^3$ *with* $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$ *be trinucleotide codes such that their associated graphs* $\mathcal{G}(\mathcal{X}_1)$ *and* $\mathcal{G}(\mathcal{X}_2)$ *satisfy the two conditions* (1) *and* (2) *of Proposition 2.1.2. Then the following statements hold:*

1. *The graph* $\mathcal{G}(\mathcal{X}_1^c)$ *where* $\mathcal{X}_1^c := \mathfrak{B}^3 \setminus \mathcal{X}_1$ *satisfies both conditions* (1) *and* (2)*;*
2. *The graph* $\mathcal{G}(Z)$ *where* $Z := \mathcal{X}_1 \cup \mathcal{X}_2$ *satisfies both conditions* (1) *and* (2)*.*

Proof. Let $\mathcal{X}_1, \mathcal{X}_2 \subseteq \mathfrak{B}^3$ with $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$ be codes such that their associated graphs $\mathcal{G}(\mathcal{X}_1)$ and $\mathcal{G}(\mathcal{X}_2)$ satisfy the two conditions (1) and (2) of Proposition 2.1.2.

Claim (1) It follows from the fact that graph $\mathcal{G}(\mathfrak{B}^3)$ satisfies the two conditions (1) and (2) of Proposition 2.1.2, and[3]

$\mathcal{G}(\mathfrak{B}^3) = \mathcal{G}(\mathcal{X}_1) \cup \mathcal{G}(\mathcal{X}_1^c)$ and $E(\mathcal{X}_1) \cap E(\mathcal{X}_1^c) = \emptyset$.

Claim (2) Condition (1) of Proposition 2.1.2 is true, since $V(Z) = V(\mathcal{X}_1) \cup V(\mathcal{X}_2)$. Let us show that Condition (2) of Proposition 2.1.2 holds as well. As $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$, it follows that $E(\mathcal{X}_1) \cap E(\mathcal{X}_2) = \emptyset$. Two cases are considered: (i) If $v \notin V(\mathcal{X}_1) \cap V(\mathcal{X}_2)$, then also $\overleftarrow{c(v)} \notin V(\mathcal{X}_1) \cap V(\mathcal{X}_2)$. Furthermore, Condition (2) of Proposition 2.1.2 is satisfied, because it holds in $\mathcal{G}(\mathcal{X}_1)$ or $\mathcal{G}(\mathcal{X}_2)$; (ii) if $v \in V(\mathcal{X}_1) \cap V(\mathcal{X}_2)$, then also $\overleftarrow{c(v)} \in V(\mathcal{X}_1) \cap V(\mathcal{X}_2)$. Additionally, Condition (2) of Proposition 2.1.2 is satisfied also, since in- and out-degrees which are equal in $\mathcal{G}(\mathcal{X}_1)$ and $\mathcal{G}(\mathcal{X}_2)$, respectively, are added.

□

Observation and data evaluation have indicated that Theorem 2.1.5 is true. However, the actual proof is not yet published and illustrates why Theorem 2.1.5 must be true.

Theorem 2.1.5. *(Theorem 3.13, [24]) Let $\mathcal{X} \subseteq \mathfrak{B}^3$ be a trinucleotide circular code of a size of at least* 18*. Then, $\mathcal{X}$ is self-complementary if, and only if,*

1. $|\ \mathcal{X}\ |$ *is even, i.e.* $|\ \mathcal{X}\ | = 18$ *or* $|\ \mathcal{X}\ | = 20$ *(and hence maximal);*
2. $V(\mathcal{X}) = \overleftarrow{c(V(\mathcal{X}))}$*;*
3. $d^+(v) = d^-(\overleftarrow{c(v)})$ *for any vertex* $v \in V(\mathcal{X})$*.*

The upcoming proof shows that Theorem 2.1.5 is true. The line of argument is structured as following: If a trinucleotide code $\mathcal{X}$ is non-self-complementary, but is nevertheless associated with a graph $\mathcal{G}(\mathcal{X})$ that satisfies the conditions (1), (2) and (3), the code has a strong non-self-complementary subcode. The proof shows that the circular permutation classes of the complements of the subcode can not be in $\mathcal{X}$. The code $\mathcal{X}$ would be either self-complementary or non-circular. Hence, it contradicts the Theorem.

[3]Recall that the union $\mathcal{G}_1 \cup \mathcal{G}_2$ of two graphs $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$ is defined as $\mathcal{G} = (V_1 \cup V_2, E_1 \cup E_2)$.

Proof. Be $\mathcal{X} \subset \mathfrak{B}^3$ circular code of size 18. The corresponding graph $\mathcal{G}(\mathcal{X})$ fulfills the Conditions (2) and (3) of the Theorem 2.1.2 but $\mathcal{X}$ is non-self-complementary. Recalling the construction of such a code, we can say that if we remove all complementary pairs from the code, the remainder must still satisfy conditions (2) and (3) of the Theorem 2.1.2. Thus, a set $\mathcal{X}^*$ constructed as in construction 2.1.3 must be a subset of $\mathcal{X}^* \subset \mathcal{X}$. Construction 2.1.3 requires that $4 \leq |\mathcal{X}^*| \leq 8$. Consequently, it can be said that:

- for each $w \in \mathcal{X}^*$ it follows that $\overleftarrow{c(w)} \notin \mathcal{X}$
- $|\mathcal{X}^*| \geq |(\alpha_1(\overleftarrow{c(\mathcal{X}^*)}) \cup \alpha_2(\overleftarrow{c(\mathcal{X}^*)}) \cap \mathcal{X}| \geq |\mathcal{X}^*| - 2$

Let $Y = \alpha_1(\overleftarrow{c(\mathcal{X}^*)}) \cup \alpha_2(\overleftarrow{c(\mathcal{X}^*)}$ be the subset of $\mathcal{X}$, which contains all codons of the circular permutation classes of $\overleftarrow{c(\mathcal{X}^*)}$, excluding the complements $\overleftarrow{c(\mathcal{X}^*)}$.

Claim (1)**:** We claim that $|Y \cap \mathcal{X}^*| < |\mathcal{X}^*| - 2$. Let us assume that $|Y \cap \mathcal{X}^*| \geq |\mathcal{X}^*| - 2$. Let $\{b_a, b_b, b_c, b_d\} = \mathfrak{B}$, where $b_c = c(b_d)$ and $b_a = c(b_b)$. One can say that for all $b_1b_2b_3 \in \mathcal{X}^*$ it follows that $b_2 \in \{b_a, b_b\}$. We denote

$$Q_1 = \{b_1b_2b_3 : b_1, b_2, b_3 \in \mathfrak{B} \wedge b_2 \in \{b_a, b_b\} \wedge b_1b_2b_3 \neq b_2b_3b_1\}$$

Hence, $|Q_1| = 30$ and $\mathcal{X}^* \subset Q_1$. Let us denote Q_2 as

$$Q_2 = \{b_1b_2b_3 : b_1b_2b_3 \in Q_1 \wedge b_1 \in \{b_a, b_b\} \vee b_3 \in \{b_a, b_b\}$$

It can be said that $|Y \cap \mathcal{X}^*| \subset Q_2$ and $|Q_2| = 22$. Further, we denote three additional subsets:

$$Q_{XYY} = \{b_1b_2b_3 : b_1b_2b_3 \in Q_1 \wedge b_1 \in \{b_a, b_b\} \dot{\vee} b_3 \in \{b_a, b_b\}$$

$$Q_{YYY} = \{b_1b_2b_3 : b_1b_2b_3 \in Q_1 \wedge b_1 \in \{b_a, b_b\} \wedge b_3 \in \{b_a, b_b\}$$

$$Q_{XYX} = \{b_1b_2b_3 : b_1b_2b_3 \in Q_1 \wedge b_1 \in \{b_c, b_d\} \wedge b_3 \in \{b_c, b_d\}$$

The sets $|Q_{XYX}| = 8$, $|Q_{YYY}| = 6$, and $|Q_{YYY}| = 16$ are listed in Table 2.3 & Table 2.2. Since Q_{YYY} consists of two circular permutation classes, it follows that $|Y \cap \mathcal{X}^* \cap Q_{YYY}| \in \{0, 2\}$. More precisely, if $|Y \cap \mathcal{X}^* \cap Q_{YYY}| = 2$, then $|\mathcal{X}^* \cap Q_{YYY}| = 2$.

We claim (i) that if $|\mathcal{X}^* \cap Q_{YYY}| = 2$, then $|\mathcal{X}^*| \geq 6$, and (ii) that if $|\mathcal{X}^* \cap Q_{XYX}| = 0$, then $|Y \cap \mathcal{X}^* \cap Q_{XYY}| = 0$.

To claim (i) let $\{b_1b_2b_3, b_4b_5b_6\} = \mathcal{X}^* \cap Q_{YYY}$, where $b_i \in \{b_a, b_b\}$ for $i = 1, \ldots, 6$

(a) if $b_2 = b_5$, it follows that $c(b_3b_2)X, c(b_6b_5)X$, $Xc(b_2b_1)$ and $Xc(b_5b_4) \in \mathcal{X}^*$, where X can be substituted with b_c or b_d.

(b) if $b_2 \neq b_5$, it follows that $b_1b_2 \neq c(b_5b_6)$ and $b_2b_3 \neq c(b_5b_4)$. Hence, $c(b_3b_2)X, c(b_6b_5)X$, $Xc(b_2b_1)$ and $Xc(b_5b_4) \in \mathcal{X}^*$, where X can be substituted with b_c or b_d.

To claim (ii) Let $|Y \cap \mathcal{X}^* \cap Q_{XYY}| = 2$ and $\{b_1b_2b_3, b_4b_5b_6\} = Y \cap \mathcal{X}^* \cap Q_{XYY}$. Without loss of generality, we may assume that $b_1, b_2, b_4, b_5 \in \{b_a, b_b\}$ and $b_3, b_6 \in \{b_c, b_d\}$. Consequently, $b_1, b_2 = c(b_4b_5)$ and $b_3 = c(b_6)$. This leaves only one code pattern:

$$\{b_ab_ab_c, b_bb_bb_d, b_bb_ab_a, b_ab_bb_b, b_db_bb_a, b_cb_ab_b\} = \mathcal{X}^*$$

An example of this code can be found in Example 15.2. In this case, $\mathcal{X}^*$ is non-circular. Therefore, $\mathcal{X}$ is non-circular, which is a contradiction. The claims (i) and (ii) prove that $|Y \cap \mathcal{X}^*| \leq |\mathcal{X}^*| - 4$. Thus, it proves claim (1).

Claim (2)**:** Let $\mathcal{G}(Y^*)$ be the associate graph to $Y^* \subset Y$, so that $|Y \cap Y^*| \geq |\mathcal{X}^*| - 2$ and $\mathcal{G}(Y^*)$ satisfies the Condition (2) and (3) of the Theorem 2.1.5. As Y^* cannot be self-complementary, it must match Construction 2.1.3. Consequently, $|\mathcal{X}^*| \geq |Y^*| \geq 4$.

(a) $Q_1 \cap Y^* = \emptyset$ and $\mathcal{X}^* \subset Q_1$. It must be true that $\mathcal{X}^* \cap \overleftarrow{c(Y^*)} = \emptyset$ and $\alpha_1(Q_{XYX}) \cup \alpha_2(Q_{XYX}) = SW(Q_{XYY})$. If $\mathcal{G}(\mathcal{X}^*)$ is acyclic and satisfies condition (2) and (3) of Theorem 2.1.5, it follows that $|\mathcal{X}* \cap Q_{XYX}| \in 0, 1, 4$. Otherwise, $\mathcal{G}(\mathcal{X}*)$ is cyclic or does not satisfy Condition (2) and (3) of Theorem 2.1.5. This is due to the fact that, if $b_cb_ab_b \in \mathcal{X}^*$, it follows that $b_ab_bb_c \notin \mathcal{X}^*$ and $b_ab_bb_c \notin \mathcal{X}^*$. Hence, for the two vertices with the labels b_c and b_d $d^+(b_c) \neq d^-(b_d)$. Therefore, without loss of generality, it can be said that there is only one possible construction of $\mathcal{X}^*$ and Y^* so that $|Y \cap Y^*| \geq |\mathcal{X}^*| - 2$:

$$\mathcal{X}^* = \{b_ab_ab_c, b_bb_bb_c, b_bb_ab_a, b_ab_bb_b, b_db_bb_a, b_db_ab_b\}$$

$$Y^* = \{b_ab_db_a, b_bb_db_b\, b_bb_cb_a, b_ab_cb_b\}$$

An example of this code $\mathcal{X}^* \cup Y^*$ is given in Example 15.3. The associated graphs $\mathcal{G}(\mathcal{X}^*)$ and $\mathcal{G}(Y^*)$ are both acyclic and both satisfy Condition (2) and (3) of Theorem 2.1.5. Yet, $\mathcal{G}(\mathcal{X}^* \cup Y^*)$ is acyclic and therefore a contradiction.

(b) $Y^* \subset Q_1$ and $\mathcal{X}^* \subset Q_1$. It is obvious that $\mathcal{X}^* \cap \overleftarrow{c(Y^*)} = \emptyset$. Furthermore, it is clear that $|(Y^* \cup \mathcal{X}^*) \cap Q_{XYX}| \leq 2$. Hence, without loss of generality, we can say that:

$$\{b_ab_ab_c, b_bb_bb_d, b_bb_ab_a, b_ab_bb_b, b_db_bb_a, b_cb_ab_b\} \subset \mathcal{X}^* \cup Y^*$$

as in claim $(1)-(ii)$, this leads to a contradiction, because $\mathcal{X}^* \cup Y^*$ is non-circular.

Claim (3)**:** Let $Y_1^* \cup \cdots \cup Y_k^* \subset \mathcal{X}^*$ so that $|Y \cap (Y_1^* \cup \cdots \cup Y_n^*)| \geq |\mathcal{X}^*| - 2$ and the associated graphs $\mathcal{G}(Y_i^*)$, where $i \in \{1, \ldots, k\}$ satisfy the Condition (2) and (3) of Theorem 2.1.5. This structure must be a contradiction. For proof see claim (2)-(b).

□

Table 2.2: Set Q_1, where all codons have a weak middle base. Put $b_2 \in \{A, T\}$ for all $b_1b_2b_3 \in Q_1$. Q_{XYX} is in the two left columns; Q_{XYY} is in the four middle columns; Q_{YYY} is in the two right columns;

CAC	GTG	CAA	TTG	GTT	AAC	AAA	TTT
CTC	GAG	CTT	AAG	GAA	TTC	ATA	TAT
CTG	CAG	CAT	ATG	GAT	ATC	AAT	ATT
GAC	GTC	CTA	TAG	GTA	TAC	TAA	TTA

Table 2.3: Set Q_1, where all codons have a strong middle base. Put $b_2 \in \{G, C\}$ for all $b_1b_2b_3 \in Q_1$. Q_{XYX} is in the two left columns; Q_{XYY} is in the four middle columns; Q_{YYY} is in the two right columns;

ACA	TGT	ACC	GGT	TGG	CCA	CCC	GGG
AGA	TCT	AGG	CCT	TCC	GGA	CGC	GCG
AGT	ACT	ACG	CGT	TCG	CGA	CCG	CGG
TCA	TGA	AGC	GCT	TGC	GCA	GCC	GGC

The following examples are constructed to illustrate the patterns used in the proof of Theorem 2.1.5.

Example 15.2. *A code $\mathcal{X}^*$ of size six so that it fits the pattern*

$$\mathcal{X}^* = \{b_ab_ab_c, b_bb_bb_d, b_bb_ab_a, b_ab_bb_b, b_db_bb_a, b_cb_ab_b\}$$

$$\mathcal{X}^* = \{AAG, ATT, CAT, GTA, TAA, TTC\}$$

is strong non-self-complementary and non-circular, but its graph $\mathcal{G}(\mathcal{X}^*)$ satisfies the Conditions (1) *and* (2) *of Proposition 2.1.2. Figure 2.5 displays the graph $\mathcal{G}(\mathcal{X})$ associated to $\mathcal{X}$.*

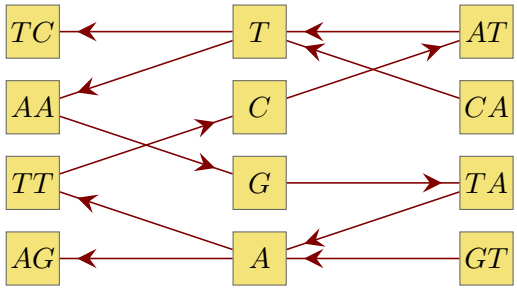


Figure 2.5: Graph $\mathcal{G}(\mathcal{X}^*)$ of the strong non-self-complementary and non-circular code $\mathcal{X}^* = \{AAG\ ,ATT,\ CAT,\ GTA,\ TAA,\ TTC\}$ of size six satisfies the two conditions (1) and (2) of Proposition 2.1.2

Example 15.3. *A code $\mathcal{X} = \mathcal{X}^* \cup Y^*$ of size ten, so that it fits the pattern*

$$\mathcal{X}^* = \{b_a b_a b_c, b_b b_b b_c, b_b b_a b_a, b_a b_b b_b, b_d b_b b_a, b_d b_a b_b\}$$

$$Y^* = \{b_a b_d b_a, b_b b_d b_b\, b_b b_c b_a, b_a b_c b_b\}$$

$$\mathcal{X} = \{AAG, ACA, AGT, ATT, CAT, CTA, TAA, TCT, TGA, TTG\}$$

is strong non-self-complementary and non-circular, but its graph $\mathcal{G}(\mathcal{X}^*)$ satisfies the two Conditions (1) *and* (2) *of Proposition 2.1.2. Figure 2.6 displays the graph $\mathcal{G}(\mathcal{X})$ associated with $\mathcal{X}$.*

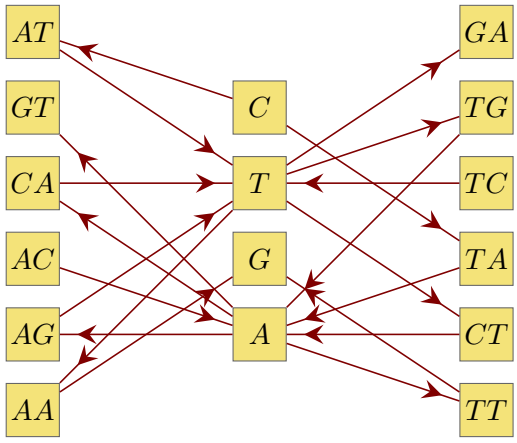


Figure 2.6: Graph $\mathcal{G}(\mathcal{X})$ of the strong non-self-complementary and non-circular code $\mathcal{X} = \{AAG,\ ACA,\ AGT,\ ATT,\ CAT,\ CTA,\ TAA,\ TCT,\ TGA,\ TTG\}$ of size ten satisfies the two conditions (1) and (2) of Proposition 2.1.2

In the subsequent section the focus moves to a more general perspective. The targets of investigation are Circular ℓ-letter codes over arbitrary alphabets Σ and comma-free codes.

2.2 Properties of circular codes in general

Most theories on the evolution of the genetic code can be grouped into one of four theory groups. In the introduction, we introduced these four theories [46, 47]. Let us briefly summarize them: The frozen accident theory, which assumes that the code has remained frozen since its origin; the stereochemical theory, based on stereochemical relationships between amino acids and specific (anti-)codons [64, 79]; the adaptive theory, proposing that the genetic code was designed to be maximally robust [76, 35]; and the coevolution theory of the genetic code, relating its evolution to amino acid biosynthetic pathways [77].

If we assume that circular codes were an influencing factor in evolution, it is important to create a hypothetical model of this process. From a biomathematical point of view, this means that we need to have improvable features as fitness values in such an evolutionary mode. To develop such features, we can use constraints of the adaptive theory as guidance. Two easily identifiable features under these constraints are the maximum size and the frameshift robustness of a code. In many respects, these characteristics contradict each other. For instance, a circular trinucleotide code has a maximal size of 20. Whereby the maximal size of the more restrictive strong comma-free trinucleotide codes is only 9, see [26]. To investigate these correlations, we fully characterise the maximal size of circular codes in section 2.2.1. Subsequently, in section 2.2.2, we present a separation of any maximal circular trinucleotide code into four equally sized comma-free codes as a hypothetical reversed development of the evolution. Lastly, section 2.2.3 delineates a mapping from circular codes over a finite alphabet onto binary codes. This method is presented as an apparatus to use methods from communication engineering, to investigate other features of circular codes in sequences.

2.2.1 Maximal size of circular codes

In the introduction section 1.1.3, some of the most important models on the evolution of the genetic code have been outlined. One of them, the adaptive theory suggesting that the genetic code was designed to be maximally robust [76, 35]. The evolution proposed in this theory focuses on minimizing the effects of errors in transcription and translation. Therefore, on the one hand, the code must have enhanced the robustness against the incorrect reading of a codon and the resulting incorporation of the wrong amino acid. On the other hand, the interest of evolu-

tion must have been to maintain or rather increase the capacity of the information processing system. If we apply this theoretical model to circular codes, however, the maximum size of such a code becomes a very essential factor in the search for hypothetical codes ancestral to the standard genetic code. Especially when we consider that it has been suggested that a code before LUCA could be based on dinucleotides or tetranucleotides [47]. The Examples 15.4, 15.5, 15.6 illustrate maximum codes of different word lengths.

Example 15.4. *This example presents a maximal circular tetranucleotide code* $\mathcal{X}_{tetra} \subset \mathfrak{B}^4$. *The maximal size of a circular tetranucleotide code is 60.*

$$\mathcal{X}_{tetra} = \{AAAC,\ AAAG,\ AAAT,\ AACC,\ AACG,\ AACT,\ AAGC,\ AAGG,\ AAGT,\ AATC,\ AATG,\ AATT,\ ACAG,\ ACAT,\ ACCC,\ ACCG,\ ACCT,\ ACGC,\ ACGG,\ ACGT,\ ACTC,\ ACTG,\ ACTT,\ AGAT,\ AGCC,\ AGCG,\ AGCT,\ AGGC,\ AGGG,\ AGGT,\ AGTC,\ AGTG,\ AGTT,\ ATCC,\ ATCG,\ ATCT,\ ATGC,\ ATGG,\ ATGT,\ ATTC,\ ATTG,\ ATTT,\ CCCG,\ CCCT,\ CCGG,\ CCGT,\ CCTG,\ CCTT,\ CGCT,\ CGGG,\ CGGT,\ CGTG,\ CGTT,\ CTGG,\ CTGT,\ CTTG,\ CTTT,\ GGGT,\ GGTT,\ GTTT\}$$

Example 15.5. *This example presents a maximal circular trinucleotide code* $\mathcal{X}_{tri} \subset \mathfrak{B}^3$. *The maximal size of a circular trinucleotide code is 20.*

$$\mathcal{X}_{tri} = \{AAC,\ AAG,\ AAT,\ ACC,\ ACG,\ ACT,\ AGC,\ AGG,\ AGT,\ ATC,\ ATG,\ ATT,\ CCG,\ CCT,\ CGG,\ CGT,\ CTG,\ CTT,\ GGT,\ GTT\}$$

Example 15.6. *This example presents a maximal circular dinucleotide code* $\mathcal{X}_{di} \subset \mathfrak{B}^2$. *The maximal size of a circular dinucleotide code is 6.*

$$\mathcal{X}_{di} = \{AC,\ AG,\ AT,\ CG,\ CT,\ GT\}$$

In order to use the circular codes as a selection factor to promote the search for these hypothetical ancestor codes, it is vital to generally characterize the behavior of these codes by adjusting the word length or cardinality of the alphabet. Thus, it is relevant to be able to explain the advantages and disadvantages of such adjustments in circular codes. This poses the following two questions:

- How does genetic information processing benefit from a circular code with increased word length?
- How does genetic information processing benefit from a circular code with an extended alphabet?

Before we answer these question we introduce an optimized algorithm to calculate the maximum size of a circular code. The obtained algorithm is faster and easier to program as the origin function to calculate the maximum size of a circular code. This facilitated working with these codes and allowed for characterizing the behavior of the maximality in terms of circular codes.

Optimization of the algorithm to calculate growth of maximality

The aim of this section is to optimize the algorithm to calculate the maximum size of circular codes. As mentioned above, this improved algorithm can be implemented easily and facilitate the research in this subject. The maximum size of a circular code depends on the word length ℓ and the cardinality of the alphabet $n = |\Sigma|$. The expression for calculating the maximum size of a circular code is based on the Möbius function $\mu(n)$, a multiplicative function in number theory. This function has values {-1,0,1} that depend on the number of prime factors of the input n. Before we show the definition of the Möbius function, we need to define $\omega(x)$. Let x be a positive integer, then $\omega(x)$ is the number of prime factors of x. Next, we define the Möbius function:

$$\mu(n) = \begin{cases} (-1)^k & \text{if } n \text{ is a square-free positive integer. } k = \omega(n), \\ 0 & \text{if } n \text{ has a squared prime factor} \end{cases}$$

Definition 16. *The different means of factorizing a number $n \in \mathbb{N}$ are denoted as follows:*

Set	***Explanation***	***Example***
P_n	*The set of prime factors of n. Including n in case n is prime.*	*If* $n = 12 \rightarrow P_n = \{3, 2\}$ *If* $n = 5 \rightarrow P_n = \{5\}$
F_n^-	*The set of all factors of n, excluding n.*	*If* $n = 12 \rightarrow F_n^- = \{6, 4, 3, 2, 1\}$
F_n	*The set of all factors of n, including n.*	*If* $n = 12$ $\rightarrow F_n = \{12, 6, 4, 3, 2, 1\}$

We define $\omega(n)$ as integer: $\omega(n) = |P_n|$

The expression to find the maximal size of a circular code for a given tuple $\{\ell, n\}$ uses the Möbius function. This function $M(n, \ell)$ subtracts the number of words in so-called incomplete circular permutation classes from the number of all words of length ℓ over a alphabet Σ with $n = |\Sigma|$ and then divides the result by ℓ. Example 16.1 demonstrates this with the calculation for dinucletide codes.

Example 16.1. *Let K_c, K_i be sets, so that K_c lists all dinucletide complete circular equivalence classes:*

$$K_c = \{AC, CA\}, \{AG, GA\}, \{AT, TA\}, \{CG, GC\}, \{CT, TC\}, \{GT, TG\}.$$

and K_i lists all dinucletide incomplete circular equivalence classes:

$$K_i = \{AA\}, \{CC\}, \{GG\}, \{TT\}$$

To calculate the maximal size of a dinucletide code where $\ell = 2$ *and* $n = |\{A, C, G, T\}| = 4$*, one has to subtract the number of words in all incomplete circular equivalence classes, denoted as* k_i

$$k_i = \sum_{K \in K_i} |K| = 4$$

from the number of all words $4^2 = 16$ *and then divide it by* $\ell = 2$*.*

$$|K_c| = \frac{n^\ell - k_i}{\ell} = \frac{4^2 - 4}{2} = 6$$

Therefore, the result of $M(n, \ell)$ is the number of complete circular permutation classes. In the article [23] it is proven that the number of complete circular permutation classes with a word length ℓ over an alphabet Σ is always equal the maximal size of circular code with a word length ℓ over Σ. Hence, there must exist a circular code over Σ with any word length ℓ, which contains one word from each complete circular permutation class. (See Examples 15.4, 15.5, 15.6)

$$M(n, \ell) = \sum_{f \in F_\ell} \mu(f) \frac{n^{\frac{\ell}{f}}}{\ell} \tag{2.1}$$

By definition, the Möbius function requires the number of all prime factors of a number. The complexity of this function (2.1) is $O(\ell^2)$. By changing the function, the complexity can be improved. Assuming that ℓ is a prime number, in this case it is trivial to set up the equation, since there are only two terms with the Möbius function as coefficient which is $\frac{\mu(1)}{\ell} \cdot n^\ell$ and $\frac{\mu(\ell)}{\ell} \cdot n$. . We simply say that $M(n, \ell) = \frac{n^\ell - n}{\ell}$ and $\omega(\ell) = 1$. Recalling the definition of $\omega(\ell)$, it can be said that if $\omega(\ell) > 1$, it follows that ℓ is not a prime number and vice versa. Therefore, it holds true that if ℓ is not a prime number, as in the trivial case, then $P_\ell = \{p_1, p_2, \ldots, p_{\omega(\ell)}\}$ is the set of all prime factors of ℓ. If we recall the definition of the Möbius function, we can say that the exponents a_i of the factors p_i in $\ell = \prod_{i=1}^{\omega(n)} {p_i}^{a_i}$ can be ignored, as only the factors in F_ℓ, where all $a_i = 1$, have an influence on the result of $M(n, \ell)$. Evidently, the highest order in the polynomial n^ℓ is always positive, since $\mu(1) = 1$. This allows the function to be expressed as $M(n, \ell) = \frac{n^\ell + z(n,\ell)}{\ell}$ as in Definition 17.

Definition 17. *Let* $M(n, \ell) = \frac{n^\ell + z(n,\ell)}{\ell}$ *be the equation to calculate the maximal size of a circular code* $\mathcal{X}$ *under an alphabet* Σ *with a word length* ℓ*, so that* $n = |\Sigma|$*.*

$$z(n,\ell) = \sum_{p_1 \in P_\ell} (-n^{\frac{\ell}{p_1}} - \sum_{p_2 \in P_\ell \setminus \{p_1\}} (-\frac{n^{\frac{\ell}{p_1 p_2}}}{2} - \sum_{p_3 \in P_\ell \setminus \{p_1,p_2\}} (-\frac{n^{\frac{\ell}{p_1 p_2 p_3}}}{3} - \\ \dots - \sum_{p_{\omega(\ell)} \in \{p_{\omega(\ell)}\}} -\frac{n^{\frac{\ell}{p_1 p_2 \cdots p_{\omega(\ell)}}}}{\omega(\ell)!})))$$

This new notation in Definition 17 allows for the reduction reducing the complexity to $2^{o(\ell)}$. The improvement is mainly achieved by ceasing the use of the Möbius function. Hence, it is no longer necessary to find and count all prime factors of all (not only prime) factors of ℓ. The Pseudocode 1 below shows an efficient way to implement the new notation of the maximal size function.

Pseudocode 1. *Function to calculate the maximal size of circular codes. The function uFactors used in the code is defined as a function that returns a unique set P of the prime factors of ℓ so that $1, \ell \notin P$.*

```
Require: ℓ, n ∈ ℕ ≥ 2
  function maxSize(n, ℓ)
    P ← uFactors(ℓ)
    E ← {1}
    CO ← {−1}
    res ← n^ℓ
    for p ∈ P do
      for idx ∈ 1 ... |E| do
        e ← E(idx) · p
        co ← CO(idx) · −1
        E.add(e)
        CO.add(co)
        res ← res + co · n^(ℓ/e)
      end for
    end for
    return result/ℓ
  end function
```

Example Procedure
Assume $\Sigma = \{0,1,2\}$, $n = |\Sigma| = 3$ *and* $\ell = 6$
$P_\ell \leftarrow \{2,3\}$
$E \leftarrow \{1\}$
$CO \leftarrow \{1\}$
$res \leftarrow 3^6 \quad \rightarrow 729$
for $p = 2$
for $e = 1 \cdot 2$, $co = 1 \cdot -1$
$E \leftarrow E \cup \{e\} \quad \rightarrow \{1,2\}$
$CO \leftarrow CO \cup \{co\} \quad \rightarrow \{1,-1\}$
$res \leftarrow 729 + co \cdot 3^3 \quad \rightarrow 702$
for $p = 3$
for $e = 1 \cdot 3$, $co = 1 \cdot -1$
$E \leftarrow E \cup \{e\} \quad \rightarrow \{1,2,3\}$
$CO \leftarrow CO \cup \{co\} \quad \rightarrow \{1,-1,-1\}$
$res \leftarrow 702 + co \cdot 3^2 \quad \rightarrow 693$
for $e = 2 \cdot 3$, $co = -1 \cdot -1$
$E \leftarrow E \cup \{e\} \quad \rightarrow \{1,2,3,6\}$
$CO \leftarrow CO \cup \{co\} \quad \rightarrow \{1,-1,-1,1\}$
$res \leftarrow 693 + co \cdot 3^1 \quad \rightarrow 696$

return $\frac{696}{6} \quad \rightarrow 116$

If we examine the maximum sizes of the Examples 15.4, 15.5, 15.6 more closely, the values seem to increase in an exponential curve in relation to the word length. By itself, this would be an indicator that word length has increased in the course of evolution. However, there are other features than the word length to consider

as factors of evolution. For instance, a maximal circular tetranucleotide code uses $6000/4^4 \approx 23.4\%$ of all possible words while the maximal circular dinucleotide code uses $600/4^2 = 37.5\%$. In the subsequent section, we refine the feature by relating the percentage of use to the circular classes.

Expression to calculate the number of all circular permutation classes

In this section, we introduce an equation $L(n, \ell)$ to calculate all circular permutation classes, not just the complete ones. This value can be used to explain the plausibility of a word length used in the evolution process. After introducing this function, Table 2.4 displays the growth of the maximal length of a code in comparison to the results of $L(n, \ell)$. The equation to obtain all circular classes for a given tuple of $\{\ell, n\}$ is slightly more complex than $M(n, \ell)$. The polynomial $M(n, \ell)$ only returns the number of complete permutation classes. A polynomial containing as well the incomplete permutation classes must be basically an extension of $M(n, \ell)$.

Proposition 2.2.1. *The polynomial $L(n, \ell)$ returns the number of all permutation classes for $\{\ell, n\}$. It is defined as:*

$$L(n, \ell) = \sum_{f \in F_\ell} (\lambda_f n^{\frac{\ell}{f}})$$

Where the coefficients λ_f for $f \in F_\ell$ are defined as:

$$\lambda_f = \frac{1}{\ell} \sum_{i \in F_f} \mu(i) \frac{f}{i}$$

Proof. The result of $M(n, \ell)$ is defined as the number of complete classes with respect to n and ℓ. Therefore, the polynomial $L(n, \ell)$ is $M(n, \ell)$ with the addition of the number of all incomplete classes. The number of incomplete classes, on the other hand, is equal to the number of complete classes of all factors of ℓ, including the number 1 but excluding ℓ. The set of factors is called F_ℓ^-. Thus, $L(n, \ell)$ can be defined as follows:

$$L(n, \ell) = \sum_{f \in F_\ell} M(n, f)$$

The set of the polynomials summed in $L(n, \ell)$ is called $L_{n,\ell}$

$$L_{n,\ell} = \{M(n, f) : f \in F_\ell^-\}$$

so that

$$L(n,\ell) = M(n,\ell) + \sum_{m \in L_{n,\ell}} m$$

Each coefficient of the proposed equation must combine all coefficients of the polynomials in $L_{n,\ell}$. The interaction is only relevant for those that contain a non-zero term with the same degree. The coefficients of each of the polynomials are 1, divided by the corresponding word length, in this case called i.

Assume $M(n, i_1)$ has a non-zero term n^x and is one of the polynomials in $L_{n,\ell}$. This leads to two conclusions: (i) $x, i_1 \in F_\ell^-$ and $x \in F_{i_1}$ and (ii) the term in $M(n, i_1)$ appears as $\frac{1}{i_1}\mu(\frac{i_1}{x})n^x$. Consequently, the coefficient $\frac{1}{i_1}\mu(\frac{i_1}{x})$ must be taken into account in the coefficient λ_f of the term n^x in $L(n, \ell)$.

Hence, $i_2 \in F_f$, so that $\frac{i_1}{x} = i_2$. In addition, for each $i_1 \in F_f$ there exists exactly one $M(n, \frac{\ell}{f1}) \in L_{n,\ell}$, which has a non-zero coefficient of the term n^x. This ensures that $\mu(i_2)$ in λ_f is equal to $\mu(\frac{i_1}{x})$ in $M(n, i_1)$. Furthermore, by the value of f one can say: $\frac{\ell}{x} = f$ and thus $\frac{f}{i_2 \cdot \ell} = \frac{1}{i_1}$.

This shows that for every $M(n, i_j)$ that has a non-zero term of degree x, there is exactly one equal element in λ_f that is the coefficient of the term n^x in $L(n, \ell)$. This applies in both directions. Each element of the sum of each coefficient λ_f appears exactly once in a coefficient of the term of equal degree in one of the polynomials in $L_{n,\ell}$. □

Table 2.4: This table displays the growth of the maximal length of a code in comparison to the results of $L(n, \ell)$. The entries contain the values for dinucleotide, trinucleotide, and tetranucleotide codes. The last column indicates the percentage use of a maximum code of all circular permutation classes.

$\lvert\Sigma\rvert$	ℓ	$L(\lvert\Sigma\rvert, \ell)$	$M(\lvert\Sigma\rvert, \ell)$	%
4	2	10	6	60%
4	3	24	20	$\approx 83\%$
4	4	60	70	$\approx 85\%$

Let us assume that the C^3 property is somehow important for the frameshift retrieval in the translation process. In a C^3 code, the circular 1-permutation as well as the circular 2-permutation of the code are again a circular code. Which means, if the code is maximal, all except for the words in incomplete classes are involved. Then, the absolute percentage of all words used in a maximal codes is of secondary importance. This implies that the values introduced in Table 2.4 are a better fitting value to measure the advantages of codes with increasing word lengths. The values show that the step between dinucleotide codes and trinucleotide words is more than

20%, while the step between trinucleotide codes and tetranucleotide codes is less than 3%.

Behavior of circular codes with increasing word length

It is true that the maximum size of circular codes increases exponentially as the word length increases. However, under restrictions, the growth of maximum size in terms of word length is not homogeneously exponential. The maximum size of a code depends on the factors of ℓ, denoted as F_ℓ (see also Section 2.2.1).

Proposition 2.2.2. *For all $n \geq 2 \in \mathbb{N}$ and $\ell \geq 2 \in \mathbb{N}$, the following must hold true:*

$$\frac{n^\ell - n^{(\lfloor \frac{\ell}{2} \rfloor + 1)}}{\ell} < M(n,\ell) \leq \frac{n^\ell - n}{\ell}$$

Proof. We claim that:

(i) The non-zero term with the highest degree in $z(n,\ell)$ (see Definition 17) has a negative coefficient,

(ii) $z(n,\ell)$ (see Definition 17) is always negative and $z(n,\ell) \leq -n$,

(iii) $-n^{(\lfloor \frac{\ell}{2} \rfloor + 1)} < z(n,\ell)$.

Summarizing claims (i), (ii) and (iii), we can say that

$$-n^{(\lfloor \frac{\ell}{2} \rfloor + 1)} < z(n,\ell) \leq -n$$

Claim (i) We claim that the term with the highest degree in $z(n,\ell)$ (see Definition 17) has a negative coefficient. Thus, $z(n,\ell)$ is negative. Suppose that P_ℓ is a set of all prime factors of ℓ. P_ℓ is defined so that $1, \ell \notin P_\ell$. Let us assume that p_l is the lowest prime factor of P_ℓ. With respect to the $M(n,\ell)$ in section 2.2.1, it can be said that $n^{\frac{\ell}{p_1}}$ is negative and $\frac{\ell}{p_1}$ is the degree of $z(n,\ell)$.

Claim (ii) $z(n,\ell)$ is always negative and equal or less than $-n$. Since the term with the highest degree has a negative coefficient, we can formulate the following expression:

$$n^{\frac{\ell}{p_1}} > \sum_{i \in F_\ell \setminus \{1, p_1\}} n^{\frac{\ell}{i}}$$

Both sides of the expression have the factor n, which allows for simplifying it:

$$n \cdot x > n \cdot y \text{ with } x, y \in \mathbb{N}$$

which follows that

$$x - 1 \geq y$$

This shows that $z(n, \ell)$ is always equal or less than $-n$.

Claim (iii) With reference to claim (i), we can say that the degree of the polynomial $z(n, \ell)$ is $\frac{\ell}{2}$ at most. Moreover, if 2 is not a factor of ℓ, it follows that the degree of the polynomial $z(n, \ell)$ is $\frac{\ell}{3}$ at most or, more generally, the degree of the polynomial is $\leq \frac{\ell}{2}$. Therefore, $n^{(\lfloor \frac{\ell}{2} \rfloor + 1)} > \sum_{i=1}^{\lfloor \frac{\ell}{2} \rfloor} n^i$, which forces that the following holds:

$$n^{(\lfloor \frac{\ell}{2} \rfloor + 1)} > |z(n, \ell)| > n \ \textit{and} \ z(n, \ell) < 0$$

In conclusion, it proves that:

$$\frac{n^\ell - n^{(\lfloor \frac{\ell}{2} \rfloor + 1)}}{\ell} < \frac{n^\ell + z(n, \ell)}{\ell} \leq \frac{n^\ell - n}{\ell}$$

Hence, it proves Proposition 2.2.2.

□

In Proposition 2.2.3, we demonstrate that $M(n, \ell)$ is monotonously increasing in respect to the word length ℓ.

Proposition 2.2.3. *If n and ℓ are integers, both are at least 2. Then, the function value $M(n, \ell) < M(n, \ell + 1)$.*

Proof. Let us assume that ℓ is prime. Therefore, $M(n, \ell+1) \gtrapprox \frac{n^{(\ell+1)} - n^{(\lfloor \frac{\ell+1}{2} \rfloor + 1)}}{\ell+1}$. In section 2.2.1, the value $z(n, \ell)$ (see Definition 17) is introduced. It is obvious that it says that $n^{(\lfloor \frac{\ell}{2} \rfloor + 1)} > |z(n, \ell)| \geq n$. It follows that in the polynomial $M(n, \ell + 1)$, the coefficient of the term of degree ℓ is zero. Therefore, we can say that the proposition must be true if the following is true:

$$\frac{n^{(\ell+1)} - n^{(\lfloor \frac{\ell+1}{2} \rfloor + 1)}}{\ell + 1} - \frac{n^\ell - n}{\ell} \geq 0$$

Since $\ell \geq 2$, it follows that this is true if

$$\ell(n^{(\ell+1)} - n^{(\lfloor \frac{\ell+1}{2} \rfloor + 1)}) - (\ell + 1)(n^\ell - n) \geq 0$$

$$n^\ell + \frac{\ell + 1}{\ell} - n^{(\ell-1)} - n^{\lfloor \frac{\ell+1}{2} \rfloor} - \frac{n^{(\ell-1)}}{\ell} \geq 0$$

It applies that for $\ell \geq 2$ one can say $\ell - 1 \geq \lfloor \frac{\ell+1}{2} \rfloor$. Thus:

$$n^{\ell} + \frac{\ell+1}{\ell} - 2n^{(\ell-1)} - \frac{n^{(\ell-1)}}{\ell} > 0$$

and further simplified

$$\frac{(n-1)n^{(\ell-1)}}{2} - \frac{n^{(\ell-1)}}{\ell} > 0$$

Which proves that

$$M(n, \ell+1) - M(n, \ell) > 0$$

□

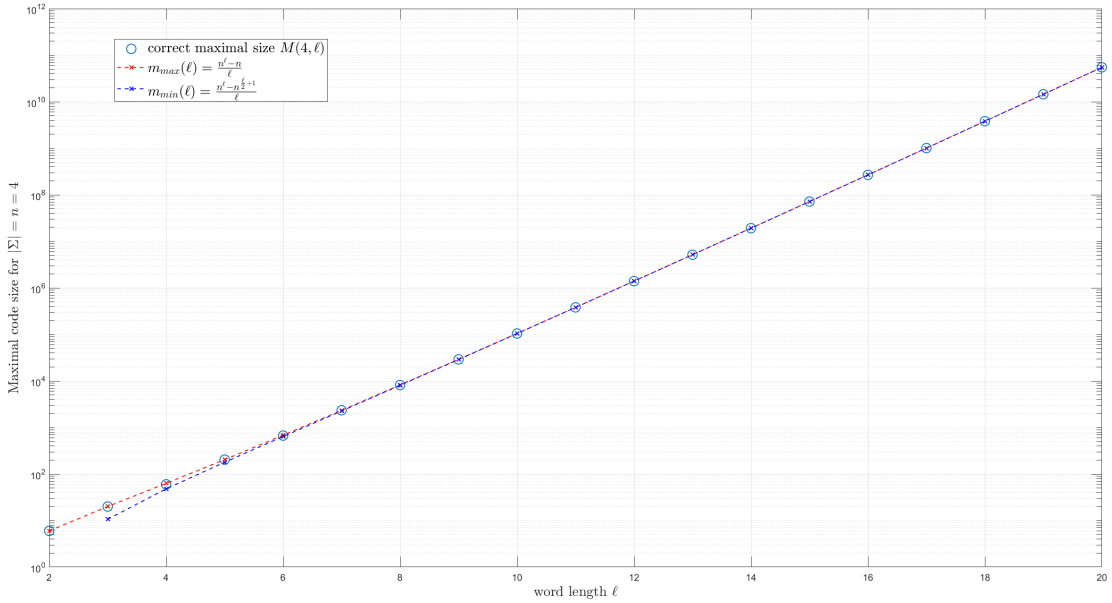


Figure 2.7: The graph shows the exponentially growing values of $M(4, \ell)$. The blue circles mark the values for $M(4, \ell)$. The upper limit $m_{max} := \frac{4^{\ell}-4^{\lfloor \frac{\ell}{2}+1 \rfloor}}{\ell}$ is represented by the red line, while the lower limit $m_{min} := \frac{4^{\ell}-4}{\ell}$ is represented by the blue line.The graphic displays the values for a range of values from 1 to 20 for ℓ.

Figure 2.7 shows the growth of the maximal size of codes over an alphabet Σ with a cardinality of 4. Without loss of generality, we can adapt this to any alphabet with a cardinality of greater or equal 2. This becomes more understandable if we include Figure 2.8.

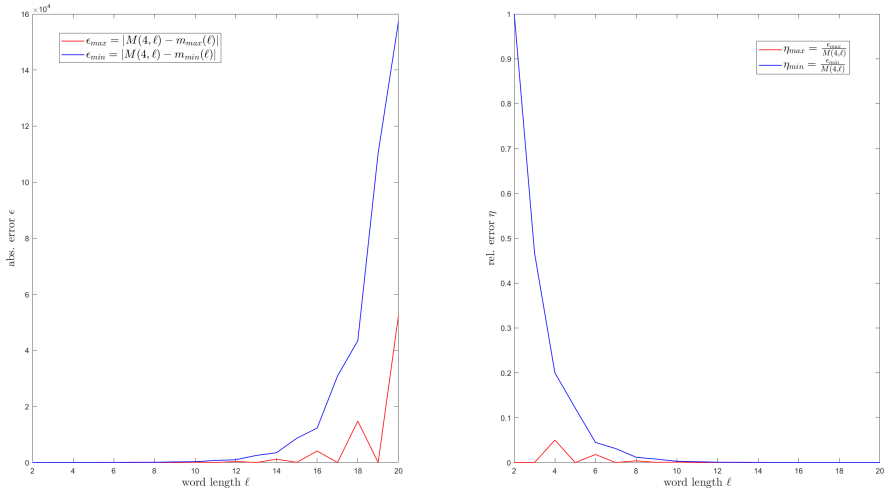


Figure 2.8: These plots depict the absolute and relative difference between $M(4,\ell)$ and its boundaries. The exponentially growing values must be always between their boundaries. It shows that $M(4,\ell)$ is always closer to its lower bound $\frac{4^\ell-4}{\ell}$ than to its upper bound $\frac{4^\ell-4^{\lfloor\frac{\ell}{2}+1\rfloor}}{\ell}$. Furthermore, it illustrates that the relative error η is close to zero.

Figure 2.7 and Figure 2.8 complete the characterization of the maximal size of circular codes. Figure 2.8 shows that the relative distance to the lower bound can be almost neglected. Therefore, it can be approximated that the first partial derivative of $M(n,\ell)$ in respect to the word length ℓ is:

$$\frac{\partial M(n,\ell)}{\partial \ell}(n,\ell) \approx n^{(-1+\ell)} log(n)$$

Despite the fixed cardinality of the alphabet used in the graphics, $\frac{\partial M(n,\ell)}{\partial \ell}$ shows that the graphics and the results can be considered as generally valid.

2.2.2 Comma-free separation of the trinucleotide permutation classes

Existing theories of hypothetical ancestor genetic codes state that there was a code, which was non-degenerate and comma-free. One of the most famous theories has been proposed by Shepherd [72], Clarke [10], Crick et al. [13]. Their theory is based on the so-called RNY code (R = purine (G or A); Y = pyrimidine (T = C); and N = (A,C,G,T)). A code following their theory would have had 16 triplets encoding only 8 amino acids. In 1990, Neveln [59] suggested that if such synchronizable codes had an influence on the development of the hypothetical ancestor code, such codes

could be used as a guideline in the search for a primitive genetic codes. We took up the idea of the article that comma-free codes can support the search for these hypothetical ancestor codes and obtained results that can support the research as a theoretical basis. In our approach, a reverse-engineering of the evolution, we present a method to separate all maximal circular codes into four equally size comma-free subsets.

To separate maximal circular codes into four comma-free subsets, this subsection presents a method for separating the 20 circular trinucleotide permutation classes into four equally sized sets. These subsets will be called comma-free subsets $\mathcal{S}_1$-$\mathcal{S}_4$. A code $\mathcal{X} \subset \mathcal{S}_i$ containing only words from one of the four comma-free subsets is either comma-free or not even circular.

Definition 18. *Let $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4 \subset \mathfrak{B}^3$ be equally sized sets of codons, so that:*

$$\mathcal{S}_i \cap \mathcal{S}_j = \emptyset \text{ with } i, j \in \{1, 2, 3, 4\},\ i \neq j$$

and

$$\bigcup_{i=1}^{i \leq 4} \mathcal{S}_i = \mathfrak{B}^3 \setminus \{AAA, TTT, CCC, GGG\}$$

Each set $\mathcal{S}_i$ is defined so that for a maximal trinucleotide circular code $\mathcal{X} \subset \mathfrak{B}^3$, the subset:

$$\mathcal{X}^S = \mathcal{X} \cap \mathcal{S}_i \text{ for } i \in \{1, 2, 4, 4\}$$

must be comma-free and $|\mathcal{X}^S| = 5$.

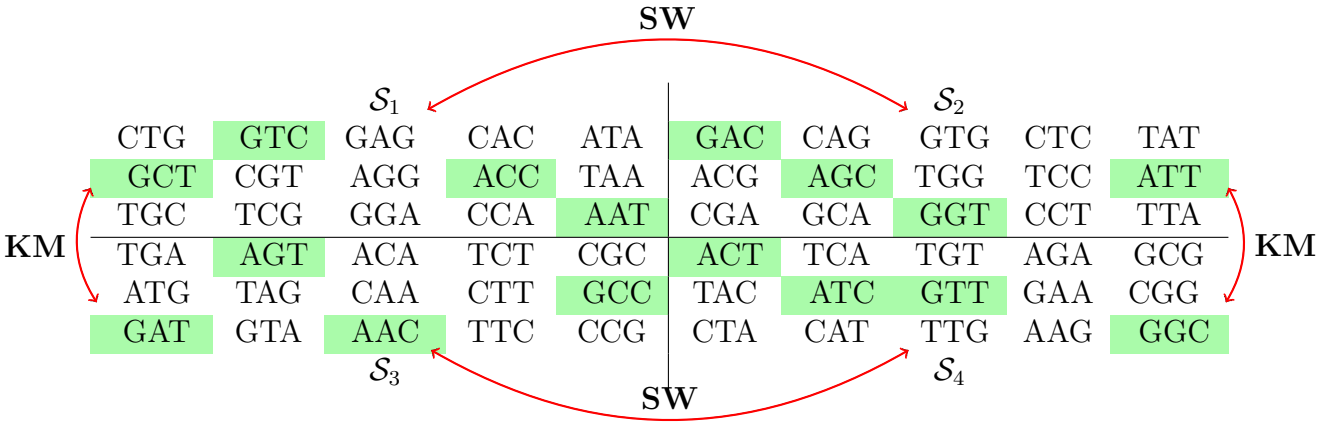


Table 2.5: $\mathcal{S}_1$-$\mathcal{S}_4$, the four comma-free subsets of $\mathfrak{B}^3$. A code $\mathcal{X} \subset \mathfrak{B}^3$ and $\mathcal{X} \subset \mathcal{S}_i$, where $i \in \{1, \ldots, 4\}$ is either comma-free or not even circular. Each subset $\mathcal{S}_i$ contains five circular permutation classes. So, if $\mathcal{X} \subset \mathcal{S}_i$ is comma-free, $|\mathcal{X}| \leq 5$ follows. The transformations $I, SW, KM, YR \in \mathcal{L}$ ($YR = KM \circ SW$) map the four subsets to each other. The green colored codons are the codons in the RNY code.

This also proves that any maximal circular code can always be separated into four comma-free codes of size five. Let us assume, $\mathcal{X}^* \subset \mathfrak{B}^3$ is a trinucleotide maximal circular code. And $\mathcal{X} = \mathcal{X}^* \cap \mathcal{S}_i$, where $i \in \{1, \ldots, 4\}$ is a subset of one of the four comma-free subsets. Therefore, $\mathcal{X}$ must be comma-free. Next, we introduce the construction of the four comma-free-subset.

Observation 2.2.1. *The four subsets in Table 2.5 separate the RNY code into four complementary subcodes of size four.*

$$\{GAT, AGT, AAC, GCC\} \leftrightarrow \{ATC, ACT, GTT, GGC\}$$

and

$$\{GCT, GTC, ACC, AAT\} \leftrightarrow \{AGC, GAC, GGT, ATT\}$$

The observation 2.2.1 presents evidence that the RNY code appears in the Table 2.5. Astonishingly the appearance of the RNY is in four equally sized complementary sets.

Construction method 2.2.4. *Recall the graph property of a graph $\mathcal{G}(\mathcal{X}) = (V(\mathcal{X}), E(\mathcal{X}))$ in association with a comma-free code. The longest path in $\mathcal{G}(\mathcal{X})$ is $l_{max}(\mathcal{X}) = 2$. To guarantee that $l_{max}(\mathcal{X}) \leq 2$, it is sufficient to guarantee that for*

dinucleotide vertices $v \in V(\mathcal{X}) \cap \mathfrak{B}^2$*, the degree* $d(v) = 1$*. Hence, if* $l_{max}(\mathcal{X}) \neq 1$*, the longest path in* $\mathcal{G}(\mathcal{X})$ *is in the form of:*

$$p : a_1 \rightarrow n_1 \rightarrow a_2$$

The four comma-free subsets $\mathcal{S}_1$*-*$\mathcal{S}_4$ *in Table 2.5 are structured so that the following applies to all codons in the same subset:*

$$b_1b_2b_3 \in \mathcal{S}_i \text{ it follows that } b_1b_2b_4, b_4b_1b_2 \notin \mathcal{S}_i \text{ where } b_3 \neq b_4$$

Note that each subset contains five complete circular permutation classes. To satisfy this condition, we reduce the permutation classes to a set of dinucleotides. Let $H = \{b_1b_2b_3, b_2b_3b_1, b_3b_1b_2\}$ *be a circular permutation class. Then, this permutation class can be transformed with a transformation* $T(\cdot)$ *into a set of the codons dinucleotide prefixes and suffixes* $T(H) = \dot{H} = \{b_1b_2, b_2b_3, b_3b_1\}$*.*

$$\text{For all } H_1, H_2 \subset \mathcal{S}_i \text{ it follows that } T(H_1) \cap T(H_2) = \emptyset$$

The comma-free subsets $\mathcal{S}_1$*-*$\mathcal{S}_4$ *can easily be constructed under the above condition. The actual construction is now trivial.*

2.2.3 Mapping functions

The focus of this last subsection of the chapter lies on the construction of a model which allows for a binary representation of a circular code over an arbitrary alphabet. To the best of our knowledge, the results obtained cannot be readily associated with biological research. This mapping is developed as an adapter to research methods from information technology. Possible applications are methods like a binomial distribution to determine if a sequence is random. Another promising branch of research are artificial neural networks to detect patterns in binary sequences.

When a code is mapped from a source Σ alphabet to a binary target $\Sigma_2 = \{0.1\}$, it is necessary to preserve the relevant properties with respect to the theory of circular code. The properties mentioned are: Circularity, comma-free, self-complementary and C^ℓ. The following section first introduces an algorithm that ignores the maximum size of the original code. A second approach shows the possible mappings where the maximum size was taken into account. However, this limits the amount of possible mappings.

We will denote $\mathcal{X}_b$ as the binary representative, Σ_2 is denoted as the binary alphabet and ℓ^* is the word length of $\mathcal{X}_b$.

Definition 19. *Let $\mathcal{X}_o \subset \Sigma^\ell$ be a ℓ-letter code over an alphabet Σ and $\mathcal{X}_b \subset \Sigma_2^{\ell_*}$ its binary representation. Then, the mapping function t:*

$$t : \Sigma^\ell \to \Sigma_2^{\ell_*}$$

is defined so that for $\mathcal{X}_b = t(\mathcal{X}_o)$ the following must hold:

- *$\mathcal{X}_b$ is circular if $\mathcal{X}_o$ and vice versa*
- *$\mathcal{X}_b$ is comma-free if $\mathcal{X}_o$ and vice versa*
- *$\mathcal{X}_b$ is self-complementary if $\mathcal{X}_o$ and vice versa*
- *$\mathcal{X}_b$ is C^ℓ if $\mathcal{X}_o$ and vice versa*

Except for the self-complementarity of the listed properties in Definition 19, all of them are naturally defined and can easily be adapted to any alphabet. To obtain the self-complementarity of the code $\mathcal{X}_o$, a complementary mapping of the letters of the alphabet Σ_2 is required. Since there is no natural complementary mapping in the binary alphabet, it must be created beforehand. For the binary case, this can be easily achieved, since there is only one logical option: $c(1) = 0$ and $c(0) = 1$.

Theorem 2.2.1. *Let $\ell \geq 2$ be an integer and $\mathcal{X}_o \subset \Sigma^\ell$ be a ℓ-letter code over an alphabet Σ. Then, the mapping function $t(\cdot)$ conserves the properties listed in Definition 19 if we find a mapping function to map the letters in the alphabet to an binary code denoted as $\mathcal{X}_w$:*

$$t_a : \Sigma \to \mathcal{X}_w$$

so that $\mathcal{X}_w = t_a(\Sigma) \subset \Sigma_2^{\ell_*/\ell}$ is a comma-free self-complementary binary (ℓ_*/ℓ)-letter code. Additionally, $\mathcal{X}_w$ must be comma-free in every circular permutation of $\mathcal{X}_w$.

Proof. Let $\mathcal{X}_o \subset \Sigma^\ell$ be a code and $\mathcal{X}_b = t(\mathcal{X}_o)$ its binary representative. We define $\mathcal{X}_w = t_a(\Sigma)$ as a binary code representing the alphabet Σ.

Circular The definition of a circular code says that any concatenation of words written on a cycle can only be decomposed into words in one reading-frame. To prove that Theorem 2.2.1 is true, we simply refer to this definition. If words can only be decomposed into words within one reading-frame, it must also be true for tuple of words in $\mathcal{X}_w$.

Comma-free If $\mathcal{X}_w$ is comma-free, this means that any concatenation of words in $\mathcal{X}_w$ is frameshift robust. It is easy to see that it must be true for tuples of $\mathcal{X}_w$.

self-complementary If $\mathcal{X}_w$ is self-complementing, it is self-evident that $\mathcal{X}_b$ is self-complementing when $\mathcal{X}_o$ is.

C^ℓ We would like to refer to the point Comma-free in this proof. Since $\mathcal{X}_w$ is defined as comma-free code in every circular permutation $\mathcal{X}_w$, we can use the same argument.

□

Construction method 2.2.5. *Let us first reuse the notation from above. Let $\mathcal{X}_o \subset \Sigma^\ell$ be a ℓ-letter code and $\mathcal{X}_b = t(\mathcal{X}_o)$ its ℓ_*-letter binary representative. We define $\mathcal{X}_w = t_a(\Sigma)$ as a (ℓ_*/ℓ)-letter binary code representing the alphabet Σ.*

We start with the construction of $\mathcal{X}_w$. In a first step, a correct word length ℓ_* of of the binary representative $\mathcal{X}_b$ needs to be determined. For this construction algorithm, the ℓ_* must be chosen as the alphabet length of the source code $\mathcal{X}_o$ plus 1 *times the word length ℓ*

$$\ell_* = \ell \cdot (|\Sigma| + 1)$$

To construct $\mathcal{X}_w$, the letters in Σ must be arranged in linear order. The positions of the complementary letters need to be inverse. Suppose $\boldsymbol{A}$, $\boldsymbol{B} \in \Sigma$ being letters in Σ and $\boldsymbol{A}$ is the complement of $\boldsymbol{B}$. Consequently, the position of $\boldsymbol{B}$ in the linear order equals 1 + ℓ minus the position of $\boldsymbol{A}$ in the order. In the genetic context we can arrange the order of the letters in eight different ways as follows: (1) $A < C < G < T$; (2) $T < C < G < A$; (3) $A < G < C < T$; (4) $T < G < C < A$; (5) $C < A < T < G$; (6) $C < T < A < G$; (7) $G < A < T < C$ or (8) $G < T < A < C$
Each linear order leads to a different mapping result.

If we then construct $\mathcal{X}_w$, we will build the words letter by letter in the predefined order. We start with a word that has a leading 1, followed by multiple 0. For each subsequent word, we replace the first 0 with a 1. This ensures that $\mathcal{X}_b$ has all the required properties. See the Example 19.1 for $\mathfrak{B}$

Example 19.1. *Let $\mathcal{X}^*$ be a trinucleotide C^3 code:*

$$\mathcal{X}^* = \{AAC,\ AAG,\ AAT,\ ACC,\ ACG,\ ACT,\ AGC,\ AGG,\ AGT,\ ATC,\ ATG,\ ATT,\ CCG,\ CCT,\ CGG,\ CGT,\ CTG,\ CTT,\ GGT,\ GTT\}$$

Let $t_a : \Sigma \to \mathcal{X}_w$ be the mapping function so that $\mathcal{X}_w \subset \Sigma_2^5$. The linear order used for $t_a(\cdot)$ is $A < C < G < T$. It can be defined as:

$$\begin{aligned} \boldsymbol{A} &\to 10000 \\ \boldsymbol{C} &\to 11000 \\ \boldsymbol{G} &\to 11100 \\ \boldsymbol{T} &\to 11110 \end{aligned}$$

Which follows that $\mathcal{X}_w = \{10000, 11000, 11100, 11110\}$. *Consequently,* $t(\mathcal{X}^*)$ *is a binary* C^{15} *code:*

$t(\mathcal{X}^*) = \{$*100001000011000, 100001000011100, 100001000011110, 100001100011000, 100001100011100, 100001100011110, 100001110011000, 100001110011100, 100001110011110, 100001111011000, 100001111011100, 100001111011110, 110001100011100, 110001100011110, 110001110011100, 110001110011110, 110001111011100, 110001111011110, 111001110011110, 111001111011110*$\}$

The Example 19.1 shows the mapping algorithm of a genetic code into a binary code. For reasons of readability, we will try to use the genetic alphabet in subsequent chapters.

Mapping with maximal size Most tuples of an alphabet Σ and a word length ℓ have a different maximum size. Only few tuples get the same result from the equation $M(n, \ell)$. Therefore, only a brute force algorithm was able to recognize the possible combinations.

#	$M(\lvert\Sigma\rvert, \ell)$	$\lvert\Sigma\rvert$	ℓ
1	3	2	4
		3	2
2	6	2	5
		4	2
3	18	2	7
		3	4
4	630	2	13
		36	2
5	6552	8	5
		27	3

Table 2.6: The table lists all possible combinations $|\Sigma|$ and ℓ with the same maximum size. The brute force algorithm checked all values of $|\Sigma| = 1, \ldots, 50$ and $\ell = 1, \ldots, 50$.

As Table 2.6 reveals, the possible combinations are limited. More disappointingly, no relationship to an evolutionary model of the genetic code could be found. Yet, this is only to the best knowledge of the author. If there are models that can be linked to these numbers, we hope that a reader will recognize this and be able to apply this table.

2.3 Summary of Chapter 2

In summary, a (genetic) circular code can be fully classified by the means of its associated graphs. We have shown that for a circular code $\mathcal{X} \subset \mathfrak{B}^3$, the path length of the longest path $l_{max}(\mathcal{X})$ must satisfy the following conditions:

1. $1 \leq l_{max}(\mathcal{X}) \leq 8$;
2. If $\mathcal{X}$ is self-complementary, then $l_{max}(\mathcal{X}) \in \{1, 2, 3, 4, 6, 8\}$, i.e., $l_{max}(\mathcal{X}) = 5, 7$ are excluded;
3. If $\mathcal{X}$ is maximal and self-complementary, then $l_{max}(\mathcal{X}) \in \{4, 6, 8\}$, i.e., in addition to (2), $l_{max}(\mathcal{X}) = 1, 2, 3$ are impossible.

These conditions are refined for maximal circular codes by the three statements below:

1. If $l_{max}(\mathcal{X}) = 4$, then the longest paths are of the form
$$a_1 \to b_1 \to a_2 \to b_2 \to a_3$$
2. If $l_{max}(\mathcal{X}) = 6$, then the longest paths are of the form
$$b_1 \to a_1 \to b_2 \to a_2 \to b_3 \to a_3 \to b_4$$
3. If $l_{max}(\mathcal{X}) = 8$, then the longest paths are of the form
$$a_1 \to b_1 \to a_2 \to \cdots \to a_4 \to b_4 \to a_5$$

In this chapter we introduced the the *reading-frame number* $n_{\mathcal{X}}$. This number indicates how many nucleotides are needed in a sequence to retrieve the reading frame. In Theorem 2.1.4, we formulated statements about the reading-frame number $n_{\mathcal{X}}$. These statements are:

1. $n_{\mathcal{X}} = l_w(p) + 2$, if $p = a_1 \to b_1 \to \cdots \to b_k$ or $p = b_1 \to a_1 \to \cdots \to a_k$;
2. $n_{\mathcal{X}} = l_w(p) + 1$, if $p = a_1 \to b_1 \to \cdots \to a_k$;
3. $n_{\mathcal{X}} = l_w(p) + 3$, if $p = b_1 \to a_1 \to \cdots \to b_k$,

where the nucleotide $b_i \in \mathfrak{B}$ and the dinucleotide $a_i \in \mathfrak{B}^2$ for any i. In the Section 2.1.2 we introduced a classification of the codes based on their longest path in the graph they are associated with. Table 2.7 lists all classes and the reading-frame numbers associated with the classes.

Longest path classes:	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Readingframe number $n_{\mathcal{X}}$:	5	6,7	8	9,10	11	12,13	14	15

Table 2.7: The reading-frame number $n_{\mathcal{X}}$ for the 8 classes of circular trinucleotide codes.

Furthermore, we were able to fully classify all graphs associated with a self-complementary code of a size of at least 18 using recognizable conditions. These conditions are formalized in the Theorem 2.1.5:

1. $\mid \mathcal{X} \mid$ is even, i.e. $\mid \mathcal{X} \mid = 18$ or $\mid \mathcal{X} \mid = 20$ (and hence maximal);
2. $V(\mathcal{X}) = \overleftarrow{c(V(\mathcal{X}))}$;
3. $d^+(v) = d^-(\overleftarrow{c(v)})$ for any vertex $v \in V(\mathcal{X})$.

Five main objectives have been achieved in the second section 2.2 of this chapter. Firstly, an advanced algorithm to calculate the maximal size of circular codes was presented. This improved algorithm is easy to script and has improved performance in terms of its input. Subsequently, an expression to calculate all circular permutation classes for an arbitrary finite alphabet and an arbitrary word length has been introduced:

$$L(n,\ell) = \sum_{f \in F_\ell} (\lambda_f n^{\frac{\ell}{f}})$$

Where the coefficients λ_f for $f \in F_\ell$ are defined as:

$$\lambda_f = \frac{1}{\ell} \sum_{i \in F_f} \mu(i) \frac{f}{i}$$

Thirdly, a full characterization of the maximal size of circular codes was given. The introduced approximated first partial derivative of M in respect of ℓ is

$$\frac{\partial M(n,\ell)}{\partial \ell} \approx n^{(-1+\ell)} log(n)$$

This was followed by a mapping function as a potential machine to replicate an evolutionary process. Table 2.6 displays feasible evolutionary transformations without loss of circular properties. Finally, the chapter concluded with a separation of the 20 circular permutation classes into four comma-free subsets, which proves that any maximal circular code can be separated into four comma-free codes of size five.

Chapter 3

Tessera circular codes

This chapter continues the examination of a potential ancestor code. The model used for the research depicted in this chapter is that of the so called Tessera code. When they first introduced this model, Gonzalez, Giannerini and Rosa [40] pointed out that the degeneration in amino acid coding has often been neglected in comprehensive evolutionary theories. Yet, there has also been some important research on this topic in the second half of the last century. In particular, the model by Yury Borisovich Rumer from 1966 [67], which was taken up by Fimmel and Strüngmann [31], was the first to show that the occurrence of degeneration as a repercussion of symmetry cannot be ignored. In the work of Gonzalez, Giannerini and Rosa a new possible evolutionary step in amino acid coding was introduced. Their idea states that degeneration is based on the principles of symmetry. Such a behavior can often be observed in nature. One of the best-known examples the field of quantum mechanics, where we can say that quantum degeneration is a consequence of symmetry. Hence, it is only logical to apply this model to the degeneration of the genetic code.

The first section 3.1 of this chapter introduces the structure and main principles of the Tessera codes. In the following chapter 3.2, we specify the behavior of Tessera codes under the constraints of the circular code theory. The sections 3.3 and 3.4 put two construction methods of circular Tessera codes forward. Both, the method in section 3.3 as well as the one in section 3.4 can construct all maximal Tessera codes. As a refined version of the method in chapter 3.3, the method in chapter 3.4 can also construct comma-free codes and circular codes of any size. Both methods conclude with properties of circular Tessera codes derived from the constructed codes.

3.1 Introduction of Tesserae

Existing theories describe an evolutionary ancestor to the modern genetic code that consisted not only of trinucleotides but also of dinucleotides, tetranucleotides or combinations of them (see [39, 3, 68, 63, 75, 78]). Among others, there are theories that propose the so-called Tessera codes, an especially selected subset of tetranucleotides, for the role of an ancestor code. Figure 3.1 shows a hypothetical evolutionary timeline including the Tessera codes. In this hypothetical model the Tessera codes appear as a link between the primitive genetic codes and the early genetic code.

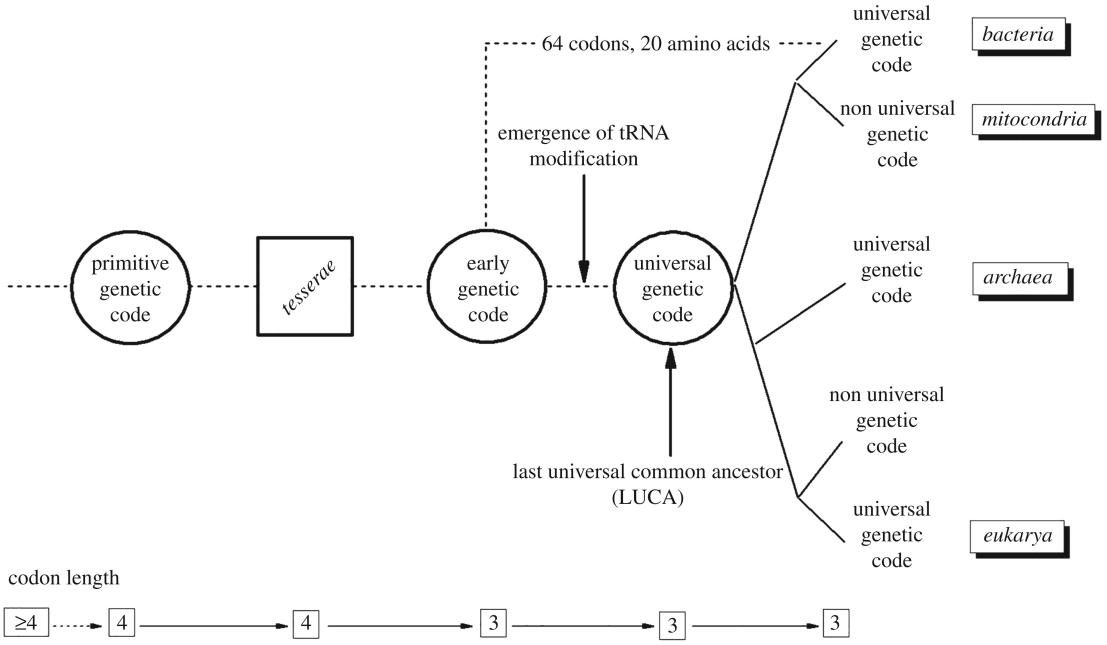


Figure 3.1: A model of the evolution of the genetic code according to the proposal of Gonzalez, Giannerini and Rosa. Each node in the evolution line represents an important milestone. The footer line shows the evolution of the word length ℓ.(Image taken from [40])

As mentioned above, the Tessera code was developed to situate symmetry as the reason for degeneration in the evolution of the genetic code. The evidence presented in article [40] illustrates that the Tessera code is one possible step of the evolutionary process of the genetic code. Therefore, a combination of Tessera code theory and circular code theory could mutually reinforce each other and also provide a hypothetical description of their roles in RNA and DNA sequences. This work serves as a guideline to identify the detection of circular Tessera codes

in prehistoric RNA/DNA sequences, thus providing a theoretical basis.

$$\mathcal{TESS} \subset \mathfrak{B}^4$$

The formal definition of Tesserae uses a Klein-four-group $(\mathcal{V}, \circ)$. In section 1.2.7 a group $(\mathcal{L}, \circ)$ has already been introduced [21]. This group is defined as $\mathcal{V} \subset \mathcal{L}$. While all transformations in $\mathcal{L}$ maintain the codon-anticodon relationship, the subset $\mathcal{V}$ can additionally be interpreted geometrically as a symmetry group of a square. $\mathcal{V} = \{I, SW, YR, KM\}$ in detail:
Identity:

$$\text{I (or id)} : (A, T, C, G) \rightarrow (A, T, C, G);$$

Strong/Weak (SW) or complementary transformation:

$$\text{SW (or c)} : (A, T, C, G) \rightarrow (T, A, G, C);$$

Pyrimidine/Purine (YR) transformation:

$$\text{YR (or p)} : (A, T, C, G) \rightarrow (G, C, T, A);$$

and Keto/Amino (KM) transformation:

$$\text{KM (or r)} : (A, T, C, G) \rightarrow (C, G, A, T).$$

These four transformations are invariant with respect to the chemical properties of nucleotides[38]. Figure 3.2 shows the geometric structure of the group $(\mathcal{V}, \circ)$. As for $(\mathcal{L}, \circ)$, $\circ$ is the associative operator and $\mathcal{V} \subset \mathcal{L}$ is a set of transformations, so that $\mathcal{V}$ contains a neutral element I and the inverted $\pi^{-1} = \pi$ for all $\pi \in \mathcal{V}$. Accordingly, all transformations in $\mathcal{V}$ are of order two. As visualized in Figure 3.2, $\mathcal{V}$ is commutative.

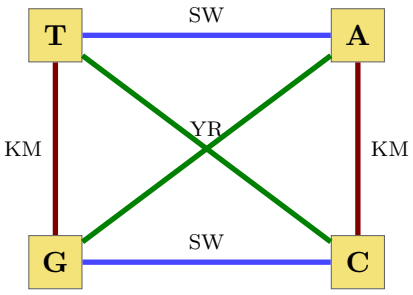


Figure 3.2: Graphical representation of the primeval base symmetries. *KM* is represented by red, *YR* by green and *SW* by blue colored lines

The following list shows three properties of $\mathcal{V}$. All of these properties can easily be traced using Figure 3.2. These properties are:

$$\pi_1 \circ \pi_1 = I \text{ for all } \pi_1 \in \mathcal{V}$$

$$\pi_1 \circ \pi_2 = \pi_3 \text{ for all } \pi_1 \neq \pi_2 \neq \pi_3 \in \mathcal{V} \setminus I$$

$$\pi_1(b) \neq b \text{ for all } \pi_1 \in \mathcal{V} \setminus I \text{ and } b \in \mathfrak{B}$$

In conclusion, these properties show that $(\mathcal{V}, \circ)$ is an isomorphism to a so called Klein-four group.

Definition 20. *There are 64 Tessera words $w \in \mathcal{TESS}$. All of them are tetranucleotides (four letter words) $\mathcal{TESS} \subset \mathfrak{B}^4$. Each word is in form of:*

$$w = b_1 b_2 \pi(b_1 b_2) \in \mathcal{TESS} \text{ with } \pi \in \mathcal{V} \text{ and } b_1 b_2 \in \mathfrak{B}^2$$

[1]

The $\mathcal{TESS}$ can be classified as a code because each concatenation of $w \in \mathcal{TESS}$ has a unique decomposition over $\mathcal{TESS}$. The following table lists all Tesserae:

Table 3.1: A table of all Tesserae with the generating transformation

Dinucleotide	I	SW	YR	KM
AA	$AAAA$	$AATT$	$AAGG$	$AACC$
CC	$CCCC$	$CCGG$	$CCTT$	$CCAA$
GG	$GGGG$	$GGCC$	$GGAA$	$GGTT$
TT	$TTTT$	$TTAA$	$TTCC$	$TTGG$
AC	$ACAC$	$ACTG$	$ACGT$	$ACCA$
AG	$AGAG$	$AGTC$	$AGGA$	$AGCT$
AT	$ATAT$	$ATTA$	$ATGC$	$ATCG$
CA	$CACA$	$CAGT$	$CATG$	$CAAC$
CG	$CGCG$	$CGGC$	$CGTA$	$CGAT$
CT	$CTCT$	$CTGA$	$CTTC$	$CTAG$
GA	$GAGA$	$GACT$	$GAAG$	$GATC$
GC	$GCGC$	$GCCG$	$GCAT$	$GCTA$
GT	$GTGT$	$GTCA$	$GTAC$	$GTTG$
TA	$TATA$	$TAAT$	$TACG$	$TAGC$
TC	$TCTC$	$TCAG$	$TCCA$	$TCGA$
TG	$TGTG$	$TGAC$	$TGCA$	$TGGT$

[1] Detailed information on the symmetrical properties of the construction of Tesserae can be found in the article [40] by Gonzalez, Giannerini and Rosa.

As can be seen in Table 3.1, there are 64 Tesserae. Consequently, each of the 64 trinucleotides $b_1b_2b_3 \in \mathfrak{B}^3$ can be uniquely extended to a valid Tessera $tess(b_1b_2b_3) = b_1b_2b_3b_4$ by determining two unique transformations $\pi_1.\pi_2 \in \mathcal{V}$ such that $b_1 = \pi_1(b_2)$ and $b_1 = \pi_2(b_3)$. This indicates that the Tessera code $\mathcal{TESS}$ is 1-error-correcting. It was shown in [32] that $\mathcal{TESS}$ can be obtained as a linear code from $\mathfrak{B}^3$ as well as by the so-called *Plotkin construction* from $\mathfrak{B}^2$ [32]. In [39] the idea of symmetric primeval adapter molecules was utilized to propose an ancient model of tRNA adapters that explains the reading mechanism and degeneracy distribution of the Tesserae. These molecules were able to recognize the normal reading-frame in the coding strand in the 3'-5' direction, in the complementary strand in the 3'-5' direction, in the coding strand in the reverse 5'-3' direction and in the complementary strand in the reverse 5'-3' direction. From the structure of the Tesserae, it is evident that self-complementary Tesserae exist, e.g. ACGT. This served as an argument in [40], which states that the Tessera code allows a degeneracy 2 and 4 only. Maintaining the degeneracy, an algorithm was suggested in [40] for passing from the Tessera code back to the (mitochondrial) genetic code in the following way: We assign a letter in the genetic alphabet via I $\leftrightarrow$ A, SW $\leftrightarrow$ T, KM $\leftrightarrow$ C and YR $\leftrightarrow$ G to each of the transformations from $\mathcal{V}$ and then perform the algorithm displayed in Figure 3.3.

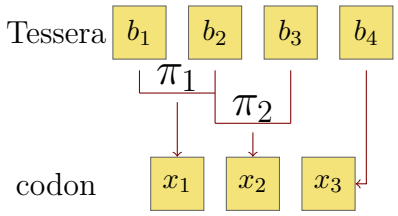


Figure 3.3: Schematic representation of the mapping function $cod(\cdot)$ from the Tessera $b_1b_2b_3b_4$ onto the codon $x_1x_2x_3$.

For instance, the Tessera $ACGT$ will be mapped onto the codon $CTT = cod(ACGT)$, since $KM(A) = C$ and $SW(C) = G$. However, note that the two mappings $tess(\cdot)$ and $cod(\cdot)$ are not inverse to each other.

3.2 Circular Tessera codes

As both the Tessera code theory and the circular code theory promote the study of the hypothetical ancestor codes, this chapter will give a theoretical guidance for the combined investigation of these two theories. In contrast to the 20 complete trinucleotide circular permutation classes, there are only twelve complete Tessera circular permutation classes, each containing four elements. Four incomplete Tessera

classes contain one element $\{AAAA\}$, $\{CCCC\}$, $\{GGGG\}$, $\{TTTT\}$, and six classes with two elements each are structured like this: $\{ACAC, CACA\}$. Table 3.2 shows all complete equivalence classes of the Tesserae.

Table 3.2: : List of complete equivalence classes. Self-complementary Tesserae are in bold.

Tessera	Shift 1	Shift 2	Shift 3	Class number
AATT	*ATTA*	**TTAA**	*TAAT*	CC_1
AAGG	*AGGA*	*GGAA*	*GAAG*	CC_2
AACC	*ACCA*	*CCAA*	*CAAC*	CC_3
CCGG	*CGGC*	**GGCC**	*GCCG*	CC_4
CCTT	*CTTC*	*TTCC*	*TCCT*	CC_5
TTGG	*TGGT*	*GGTT*	*GTTG*	CC_6
AGCT	*GCTA*	**CTAG**	*TAGC*	CC_7
TGCA	*GCAT*	**CATG**	*ATGC*	CC_8
GTAC	*TACG*	**ACGT**	*CGTA*	CC_9
AGTC	*GTCA*	*TCAG*	*CAGT*	CC_{10}
TCGA	*CGAT*	**GATC**	*ATCG*	CC_{11}
ACTG	*CTGA*	*TGAC*	*GACT*	CC_{12}

In the following, the set of 48 Tesserae in Table 3.2 will be called $\mathcal{TE} \subset \mathcal{TESS}$. Recalling the argument used for the construction of the equation $M(|\Sigma|, \ell)$ (see Section 2.2.1), the maximal size of a code is equal to the number of complete circular permutation classes. Hence, the maximal size of a circular Tessera code is 12. Note, that $M(4, 4) = 60$ (for all tetranucleotide), whereby the maximal size of a circular Tessera code is only 12 [28].

Definition 21. *The set of 48 Tesserae $\mathcal{TE}$ of the twelve complete Tessera circular permutation classes in Table 3.2 will be denoted as*

$$\mathcal{TE} \subset \mathcal{TESS}$$

Definition 22. *A circular Tessera code is called maximal if it contains exactly twelve words.*

The calculations below use the representation of the codes as a graph, in particular, the components of the graphs associated with Tessera codes. Recalling the graph component construction, a Tessera code must have two components: the 1-component $\mathcal{C}_1(\cdot)$ and the 2-component $\mathcal{C}_2(\cdot)$. The 1-component contains only 1-nodes and 3-nodes, while the 2-component contains all 2-nodes (see Definition 10).

Proposition 3.2.1. *Let $\mathcal{X}$ be a Tessera code. Then the following holds:*

(i) *The maximal length of a cycle in the 1-component $\mathcal{C}_1(\mathcal{X})$ is 2*

(ii) *The maximal length of a path in the 1-component $\mathcal{C}_1(\mathcal{X})$ that does not contain a cycle is also 2;*

(iii) *The maximal length of a cycle in 2-component $\mathcal{C}_2(\mathcal{X})$ is 4. In particular, the maximal length of a path that does not contain a cycle is 3.*

Proof. Let $\mathcal{X}$ be a Tessera code. We first prove (i) by showing that any path in $\mathcal{C}_1(\mathcal{X})$ of length 2 which starts with an 1-node must contain a cycle. Hence, assume that $\mathcal{C}_1(\mathcal{X})$ contains a path of length 2, e.g.

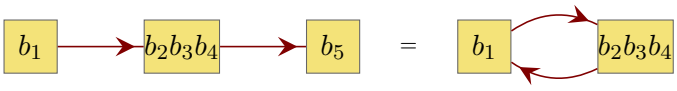


Figure 3.4: A path from nucleotide to nucleotide in a $\mathcal{C}_1(\mathcal{X})$. It follows that $\mathcal{X}$ is a non-circular Tessera code

According to Figure 3.4, one can say that $b_1b_2b_3b_4$ and $b_2b_3b_4b_5$ are valid Tesserae in $\mathcal{X}$. By the definition of Tesserae, there is a transformation $\pi \in \mathcal{V}$ such that $\pi(b_2) = b_4$ and $\pi(b_3) = b_1$. However, the definition also implies that $\pi(b_3) = b_5$ and so $b_1 = b_5$. This confirms that $\alpha_1(b_1b_2b_3b_4) = b_2b_3b_4b_1 \in \mathcal{X}$ and, consequently, that Figure 3.4 must be true.

Next, we prove (ii) by showing that a path of length 2 in $\mathcal{C}_1(\mathcal{X})$ may not contain a cycle if it starts with a trinucleotide. Assume that $\mathcal{C}_1(X)$ contains a path of length 2 that begins with a 3-node, e.g.

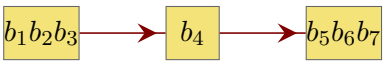


Figure 3.5: A path from trinucleotide to trinucleotide in a $\mathcal{C}_1(\mathcal{X})$.

According to Figure 3.5 one can say that $b_1b_2b_3b_4$ and $b_4b_5b_6b_7$ are valid Tesserae in $\mathcal{X}$. As in claim (i) the path can be circular if $\alpha_3(b_1b_2b_3b_4) = b_4b_5b_6b_7 \in \mathcal{X}$. Nevertheless, $b_1b_2\pi_1(b_1b_2)$ and $b_4b_5\pi_2(b_6b_7)$ where $\pi_1 \neq \pi_2 \in \mathcal{V}$ are also valid Tesserae. Hence, the path is not cyclic.

Claims (i) and (ii), however, show that the path of length 3 in $\mathcal{C}_1(\mathcal{X})$ must be cyclic.

We now prove (iii) by showing that any path of length 4 in $\mathcal{C}_2(\mathcal{X})$ contains a cycle. Assume that $\mathcal{C}_2(\mathcal{X})$ contains a path of length 4, e.g.

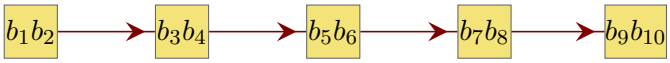


Figure 3.6: Let $\mathcal{X} \subset \mathcal{TE}$ be a Tessera code and $\mathcal{C}_2(\mathcal{X})$ the 2-component in the associated graph. According to this figure, the Tesserae $b_1b_2b_3b_4$, $b_3b_4b_5b_6$, $b_5b_6b_7b_8$ and $b_7b_8b_9b_{10}$ must be in $\mathcal{X}$. The figure illustrates the resulting path of size four in $\mathcal{C}_2(\mathcal{X})$.

By definition of $\mathcal{G}(\mathcal{X})$, there are permutations $\pi_1, \pi_2, \pi_3, \pi_4 \in \mathcal{V}$ such that

$$\pi_1(b_1b_2) = b_3b_4, \pi_2(b_3b_4) = b_5b_6, \pi_3(b_5b_6) = b_7b_8, \pi_4(b_7b_8) = b_9b_{10}$$

If one of the π_i is the identity, we obtain a cycle of length 1 (a loop). Thus, all π_i are different from the identity. If $\pi_1 = \pi_2$, then $b_1b_2 = b_5b_6$, since $\pi_1^2 = I$. Thus, this presents us with a cycle of length 2. Consequently, $\pi_1 \neq \pi_2$ and similarly $\pi_2 \neq \pi_3$, $\pi_3 \neq \pi_4$. If $\pi_1 \neq \pi_3$, then the group structure of $\mathcal{V}$ implies that $\pi_1 \circ \pi_2 = \pi_3$ and so $b_7b_8 = b_1b_2$. Hence, we obtain a cycle of length 3. Finally, if $\pi_1 = \pi_3$, then similar arguments to those listed above show that we get a cycle of length 3 or $\pi_2 = \pi_4$ holds. Now,

$$\pi_4(\pi_3(\pi_2(\pi_1(b_1b_2)))) = \pi_2(\pi_1(\pi_2(\pi_1(b_1b_2)))) = b_9b_{10}$$

,

but $\mathcal{V}$ is commutative, and all elements in $\mathcal{V}$ are of order 2. Hence,

$$b_9b_{10} = \pi_2(\pi_1(\pi_2(\pi_1(b_1b_2)))) = b_1b_2.$$

Consequently, the path itself is a cycle of length 4. As a corollary, we obtain an important theorem. [2]

□

Proposition 3.2.1 can be interpreted so that Tessera codes favor circularity. Short paths in the representing graphs equal small reading-frame numbers. In the following section, we construct circular and comma-free Tessera codes in order to identify characteristic properties of such codes.

3.3 Construction of circular Tessera codes

This section proposes a method to construct all circular Tessere codes. The first part of the construction method, which is also published in the paper [28], focuses on a general construction of circular Tessera codes. The second part depicted in

[2]Note, that part (ii) was also obtained in a bachelor thesis [9] with a much more technical proof.

the subsequent chapter 3.4 is unpublished and refines this construction. Based on the refined algorithm, a combinatorial calculation is introduced. This allows to calculate the exact number of codes with certain properties.

3.3.1 Construction using equivalence classes of dinucleotides

The construction of all maximum circular Tessera codes, which is introduced in this chapter, is achieved by following three main steps. First, the 16 possible dinucleotides are divided into four equivalence classes of size four by the means of the Klein-four-group $\mathcal{V}$. Secondly, for each of the four equivalence classes obtained, we define a tournament, *i.e.* a directional graph the distinctive features of four vertices representing each dinucleotide. We use these tournaments to construct non-circular codes of length 24 with an acyclic 2-component $\mathcal{C}_2$. Finally, in a third step, we construct maximal circular Tessera codes from the codes obtained in the previous step. The last section of this chapter is dedicated to the properties obtained from the constructed codes.

Step 1: As was already proven, the graph 1-component $\mathcal{C}_1(\mathcal{X})$, which is associated with a circular Tessera code $\mathcal{X} \subset \mathcal{TE}$, either has no path greater than 2 or $\mathcal{X}$ is non-circular. More precisely, if $\mathcal{C}_1(\mathcal{TE})$ is acyclic, the code $\mathcal{TE}$ must contain two Tesserae from the same circular permutation class. Considering this, the construction of a maximal circular Tessera code can nearly be reduced to the problem of constructing a valid and acyclic $\mathcal{C}_2(\mathcal{X})$ that represents a valid Tessera code $\mathcal{TE}$.

In these first two steps, we will show how to construct these acyclic maximal $\mathcal{C}_2$. The codes associated with the $\mathcal{C}_2$ allow to derive all maximum circular Tessera codes. To start the construction of the $\mathcal{C}_2$, the 16 dinucleotides $\mathfrak{B}^2$ must be divided into four equivalence classes of four dinucleotides. We refer to them as:

$$\mathfrak{B}_I \cup \mathfrak{B}_{SW} \cup \mathfrak{B}_{YR} \cup \mathfrak{B}_{KM} = \mathfrak{B}^2$$

Each of these equivalence classes is defined as an orbit of the dinucleotides $w \in \mathfrak{B}^2$ under the transformations in $\mathcal{V}$. The orbit of an element w is denoted as $\mathcal{V} \cdot w$:

$$\mathcal{V} \cdot w = \{g \cdot w \mid g \in \mathcal{V}\}$$

When $\mathcal{V}$ acts on $\mathfrak{B}^2$, we get four orbits of size four. Each orbit represents an equivalence class. It can be observed that for all dinucleotides b_1b_2 in the same equivalence class, a transformation $\pi \in \mathcal{V}$ maps the first nucleotide onto the second $b_1 = \pi(b_2)$. Table 3.3 below shows the four equivalence classes.

$\mathfrak{B}_I$	$\mathfrak{B}_{SW}$	$\mathfrak{B}_{YR}$	$\mathfrak{B}_{KM}$
AA	AT	AC	AG
TT	TA	TG	TC
CC	CG	CA	CT
GG	GC	GT	GA

Table 3.3: Each column shows one of the four equivalence classes of dinucleotides. For each class, the same transformation applied to the first nucleotide of each dinucleotide also yields the second. The column headings are the names of the equivalence classes. The header index is the unique transformation used to map the first nucleotide of a dinucleotide onto the second nucleotide.

Proposition 3.3.1. *Each Tessera $b_1b_2b_3b_4$ can be uniquely mapped to a tuple (b_1, β, γ) where $b_1 \in \mathfrak{B}$ and $\beta, \gamma \in \mathcal{V}$.*

Proof.

$$\beta \in \mathcal{V}, \gamma \in \mathcal{V} \setminus I, b_1 \in \{A, T, G, C\}$$

$$b_1b_2b_3b_4 = b_1b_2\gamma(b_1b_2) \text{ and } b_1 = \beta(b_2)$$

follows that:

$$b_2 = \beta(b_1) \ \& \ b_3 = \gamma(b_1) \ \& \ b_4 = (\beta \circ \gamma)(b_1)$$

□

Proposition 3.3.1 presents a proof that $\pi_1(b_1) = \pi_1(b_2)$ and $\pi_1(b_3) = \pi_1(b_4)$ for any Tessera $b_1b_2b_3b_4 \in \mathcal{TE}$ and $\pi_1 \in \mathcal{V}$. The following Proposition 3.3.2 uses these circumstances.

Proposition 3.3.2. *For each Tessera $b_1b_2b_3b_4$ the two dinucleotides b_1b_2 and b_3b_4 belong to the same equivalence class.*

Proof. Each Tessera can be displayed as:

$$b_1b_2 = \pi_1(b_3b_4) \text{ with } \pi_1 \in \mathcal{V}.$$

According to Proposition 3.3.1, this entails that:

$$b_1\pi_2(b_2) \leftrightarrow b_3\pi_2(b_4) \text{ with } \pi_2 \in \mathcal{V}$$

This simply shows that b_1b_2 and b_3b_4 belong to the same equivalence class of table 3.3. □

Definition 23. *Hence, all 48 circular Tesserae $\mathcal{TE}$ can be separated into four subsets of size 12:*

$$\mathcal{TE} = \mathcal{TE}_I \cup \mathcal{TE}_{SW} \cup \mathcal{TE}_{YR} \cup \mathcal{TE}_{KM}$$

according to the equivalence classes.

$$\mathcal{TE}_\beta = \{D_1 D_2 \in \mathfrak{B}^4 | D_1, D_2 \in \mathfrak{B}_\beta\} \text{ where } \beta \in \mathcal{V}^3$$

Therefore, each Tessera code $\mathcal{X}$ must be decomposable into four code fragments with respect to the four Tessera subsets. These code fragments are called $\mathcal{X}_I$, $\mathcal{X}_{SW}$, $\mathcal{X}_{YR}$ and $\mathcal{X}_{KM}$. In Definition 24, the fragments are correctly defined:

Definition 24. *Let's assume $\mathcal{X} \subset \mathcal{TE}$ is a circular Tessera code. Then $\mathcal{X} = \mathcal{X}_I \cup \mathcal{X}_{SW} \cup \mathcal{X}_{YR} \cup \mathcal{X}_{KM}$ with:*

$$\mathcal{X}_I \subseteq \mathcal{TE}_I$$

$$\mathcal{X}_{SW} \subseteq \mathcal{TE}_{SW}$$

$$\mathcal{X}_{KM} \subseteq \mathcal{TE}_{KM}$$

$$\mathcal{X}_{YR} \subseteq \mathcal{TE}_{YR}$$

It follows that each $\mathcal{C}_2$ associated with the Tessera code must be divisible into four disjoint components. Each of these components only consists of nodes representing the dinucleotides of only one of the equivalence classes listed in Table 3.3.

In graph theory, a tournament is defined as a directed graph so that the following holds: Every complete directed acyclic graph must be a tournament. Figure 3.7 shows such a tournament. The number of edges in a tournament is $\binom{n}{2}$, where n is the number of vertices of the graph. Thus, if $\mathcal{X}_* \in \{\mathcal{X}_I, \mathcal{X}_{SW}, \mathcal{X}_{YR}, \mathcal{X}_{KM}\}$ and if $\mathcal{C}_2(\mathcal{X}_*)$ is acyclic, then $\mathcal{C}_2(\mathcal{X}_*)$ has at most $\binom{4}{2} = 6$ edges. Furthermore, it can be summarized that for each circular Tessera code, $\mathcal{X}$ can be decomposed as follows:

$$\mathcal{X} = \mathcal{X}_I \cup \mathcal{X}_{SW} \cup \mathcal{X}_{YR} \cup \mathcal{X}_{KM}$$

$$\text{where } |\mathcal{X}_I| \leq 6, |\mathcal{X}_{SW}| \leq 6, |\mathcal{X}_{YR}| \leq 6, |\mathcal{X}_{KM}| \leq 6$$

[3] Let's assume $b_1, b_2, b_3, b_4 \in \{A, T, C, G\}$ so that $b_1 b_2 b_3 b_4$ is a Tessera. D_1 and D_2 are denoted as the two dinucleotide elements. ($D_1 = b_1 b_2$ and $D_2 = b_3 b_4$)

Step 2: Construction of codes assigned to a maximal acyclic $\mathcal{C}_2(\mathcal{X}')$. Based on the fact that each Tessera code can be decomposed into the four fragments, it is possible to construct four maximal (size of six) circular fragments $\mathcal{X}_i$, $\mathcal{X}_{SW}$, $\mathcal{X}_{YR}$ and $\mathcal{X}_{KM}$ as circular Tessera codes to obtain a code $\mathcal{X}'$ of length 24. To ensure that $\mathcal{C}_2(\mathcal{X}')$ is an acyclic code, each fragment must be represented by an acyclic $\mathcal{C}_2$. Fortunately, it can be said that if each fragment has an acyclic $\mathcal{C}_2$, then $\mathcal{C}_2(\mathcal{X}')$ must also follow. This must be true, since the $\mathcal{C}_2$ components in a graph representing the fragments are disjointed. So far, we can say that $\mathcal{X}'$ must be structured like this:

$$\mathcal{X}' = \mathcal{X}_I \cup \mathcal{X}_{SW} \cup \mathcal{X}_{YR} \cup \mathcal{X}_{KM} \text{ with } |\mathcal{X}'| = 24$$

and

$$|\mathcal{X}_I| = |\mathcal{X}_{SW}| = |\mathcal{X}_{YR}| = |\mathcal{X}_{KM}| = 6$$

The $\mathcal{C}_2(\mathcal{X}')$ is nothing other than the composition of the four disjoint components. Hence, $\mathcal{C}_2(\mathcal{X}')$ must be acyclic if each component is acyclic. Since a circular Tessera code has a size of 12 at the most, $\mathcal{G}(\mathcal{X}')$ must not be acyclic. Therefore, $\mathcal{C}_1(\mathcal{X}')$ must not be acyclic. Yet, for this step it can be ignored that the $\mathcal{C}_1(\mathcal{X}')$ will be cyclic, since the design goal is to obtain an acyclic $\mathcal{C}_2(\mathcal{X}')$. As shown above, it is sufficient that $\mathcal{C}_2(\mathcal{X}_I)$, $\mathcal{C}_2(\mathcal{X}_{SW})$, $\mathcal{C}_2(\mathcal{X}_{YR})$ and $\mathcal{C}_2(\mathcal{X}_{KM})$ are acyclic to ensure that $\mathcal{C}_2(\mathcal{X}')$ is acyclic. Thus, each component must be an isomorphism to the graph in Figure 3.7. As mentioned above, a complete directed acyclic graph must be a so called tournament.

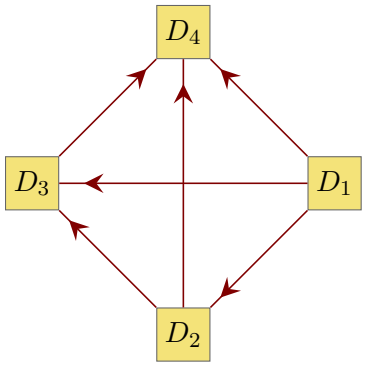


Figure 3.7: A tournament with four vertices.

Construction of a tournament: To construct such an isomorphism, one of the numbers 1, 2, 3, or 4 can be assigned to the vertices representing the dinucleotides and add directional edges from each vertex to the vertices with a higher number. This leads to 4! possible assignments per subgraph. Consequently,

there are a total of $(4!)^4 = 331776$ Tessera codes of size 24 with an acyclic $\mathcal{C}_2$. The representatives of these codes will be called $\mathcal{X}'$.

Step 3: The following step uses the 331776 Tessera codes to construct all possible maximal circular Tessera codes. The $\mathcal{C}_2$ is already acyclic for all these codes. Hence, it is necessary to focus on the $\mathcal{C}_1$. It has been shown that each cyclic path in $\mathcal{C}_1$ must be associated with two Tesserae from the same permutation class. Thus, since there are twelve complete circular permutation classes, it follows that each $\mathcal{C}_1$ representing one of the constructed Tessera codes has at least twelve cyclic paths of length 2. The subsequent section explains the nature of these cyclic paths and illustrates how they can be removed. In order to do this, the structure of the subsets of $\mathcal{TE}$ must first be explained in detail.

Proposition 3.3.3. *Let $\mathcal{TE}_\beta, \mathcal{TE}_\gamma \in \{\mathcal{TE}_I, \mathcal{TE}_{SW}, \mathcal{TE}_{YR}, \mathcal{TE}_{KM}\}$ be two subsets of $\mathcal{TE}$ and $\beta, \gamma \in \mathcal{V}$. Then the following must hold:*

(i) The set of the circular 2-permutation of all words in $\mathcal{TE}_\beta$ is again $\mathcal{TE}_\beta$.

$$\mathcal{TE}_\beta = \alpha_2(\mathcal{TE}_\beta)$$

(ii) The set of the circular 1-permutation of all words in $\mathcal{TE}_\beta$ is the set of the circular 3-permutation of all words in $\mathcal{TE}_\beta$ and vice versa.

$$\alpha_3(\mathcal{TE}_\beta) = \alpha_1(\mathcal{TE}_\beta) \text{ and } \alpha_1(\mathcal{TE}_\beta) = \alpha_3(\mathcal{TE}_\beta)$$

(iii) The set of the circular 1-permutation of all words in $\mathcal{TE}_{\beta\circ\gamma}$ and $\mathcal{TE}_{(\beta)}$ have an intersection of four words.

$$|\mathcal{TE}_\gamma \cap \alpha_1(\mathcal{TE}_{(\beta\circ\gamma)})| = 4$$

Proof. First, we prove claim (i). Let $b_1b_2b_3b_4 \in \mathcal{TE}_\beta$ be a Tessera and $\beta \in \mathcal{V}$. Assertion (i) can be proven rather easily because for every $b_1b_2b_3b_4 \in \mathcal{TE}_\beta$ it follows that $b_1b_2, b_3b_4 \in \mathfrak{B}_\beta$. This must be true. Consequently, $b_3b_4b_1b_2 \in \mathcal{TE}_\beta$.

Next, we prove claim (ii). If (i) is true, (ii) follows. This can be confirmed with the associative behavior of the circular permutation $(\alpha_1 \circ \alpha_2)(\cdot) = \alpha_3(\cdot)$ and $(\alpha_3 \circ \alpha_2)(\cdot) = \alpha_1(\cdot)$.

Finally, we prove claim (iii). Let $b_1b_2, \gamma(b_1b_2) \in \mathfrak{B}_\beta$ be two dinucleotides in the same equivalence class where $\beta \in \mathcal{V}$ and $\gamma \in \mathcal{V} \setminus I$. The claim requires that each circular 1-permutation of $\alpha_1(b_1b_2\gamma(b_1b_2)) = b_2\gamma(b_1b_2)b_1$ is in $\mathcal{TE}_{(\beta\circ\gamma)}$. This must

be true, since $b_2(\beta \circ \gamma)(b_1)$ and $b_1(\beta \circ \gamma)(b_2) \in \mathfrak{B}_{(\beta\circ\gamma)}$. Hence, there are exactly four Tessera in form of $D\gamma(D) \in \mathcal{TE}_\beta$ for all $D \in \mathfrak{B}_\beta$. It follows that only these four Tesserae in $\mathcal{TE}_\beta$ have their circular 1-permutation in $\mathcal{TE}_{(\beta\circ\gamma)}$. □

The structure outlined in Propostion 3.3.3 implies that the twelve Tesserae in each of the four subsets can be classified in three groups of four Tessera. Each group consists of two Tesserae and their circular 2-permutations. These groups are united by the fact that all 1-permutations and 3-permutations are in the same different subsets.

Proposition 3.3.4. *Let $\mathcal{X}'$ be one of the codes of size 24. For any Tessera $w \in \mathcal{X}'$, either $\alpha_1(w)$ or $\alpha_3(w)$ is in $\mathcal{X}'$.*

Proof. Therefore, let us assume the following:

$$\beta, \epsilon \in \mathcal{V}, \gamma \in \mathcal{V} \setminus I, \epsilon = \beta \circ \gamma, b_1 \in \{A, T, G, C\}$$

so that:

$$\mathfrak{B}_\beta, \mathfrak{B}_\epsilon \in \{\mathfrak{B}_I, \mathfrak{B}_{SW}, \mathfrak{B}_{YR}, \mathfrak{B}_{KM}\} \text{ and } \mathfrak{B}_\beta \neq \mathfrak{B}_\epsilon$$

with:

$$b_2 = \beta(b_1), b_3 = \gamma(b_1) = \epsilon(b_2), b_4 = \epsilon(b_1)$$

$$w_1 = b_1b_2b_3b_4 = b_1b_2\gamma(b_1b_2)$$

which shows that:

$$w_1 = b_1b_2b_3b_4 \text{ and } \alpha_2(w_1) = b_3b_4b_1b_2 \rightarrow b_1b_2, b_3b_4 \in \mathfrak{B}_\beta$$

$$\alpha_3(w_1) = b_4b_1b_2b_3 \text{ and } \alpha_1(w_1) = b_2b_3b_4b_1 \rightarrow b_2b_3, b_4b_1 \in \mathfrak{B}_\epsilon$$

The two words w_1 and $\alpha_2(w_1)$ are represented by two directed edges between b_1b_2 and b_3b_4 in the $\mathcal{C}_2(\mathcal{TE}_\beta)$. Hence, only one of them can be in $\mathcal{X}'$. Thus:

$$|\mathcal{X}' \cap \mathcal{TE}_\gamma \cap \alpha_1(\mathcal{TE}_\epsilon)| = 2$$

The two other words of the same permutation class $\alpha_1(w_1)$ and $\alpha_3(w_1)$ are represented by the two directed edges between b_2b_3 and b_4b_1 in the $\mathcal{C}_2(\mathcal{TE}_\epsilon)$. Figures 3.9-3.12 illustrate these dependencies. If this is applied to the constructed Tessera codes of size 24, it can be said that every word in such a code $\mathcal{X}'$ has a circular equivalent within the same code $\mathcal{X}'$.

$$\text{Assuming } w \in \mathcal{X}', \textit{ it follows that } \alpha_1(w) \in \mathcal{X}' \dot{\vee} \alpha_3(w) \in \mathcal{X}'$$

This must be true because, for one, every tournament is completed. Consequently, for each Tessera in $w \in \mathcal{TE}'$ at least one of the two $w \in \mathcal{X}'$ or $\alpha_2(T) \in \mathcal{X}'$.

Additionally, each tournament is acyclic. Thus, no more than one of the two $w \in \mathcal{X}'$ or $\alpha_2(w) \in \mathcal{X}'$. The same must apply to the so-called circular equivalence words $\alpha_1(w)$ and $\alpha_3(w)$. It needs to be noted, however, that they must be represented in a tournament other than w and $\alpha_2(w)$.

□

From the proof given above, it can be seen that:

$$|\mathcal{X}' \cap \alpha_1(\mathcal{X}')| = |\mathcal{X}' \cap \alpha_3(\mathcal{X}')| = 12$$

and

$$|\alpha_2(\mathcal{X}') \cap \mathcal{X}'| = 0$$

To ensure that the codes are circular, one of each circular equivalent pairs must be removed. This must be done for all twelve circular equivalent word pairs in such a code $\mathcal{X}'$. It follows that any of the 331776 codes can be used to construct 2^{12} circular codes. Some of the codes can be constructed more than once. Thus, the total amount of $2^{12} \times (4!)^4 = 1358954496$ constructed maximum circular Tessera codes contains all maximum circular Tessera codes at least once. Nevertheless, some codes appear more than once.

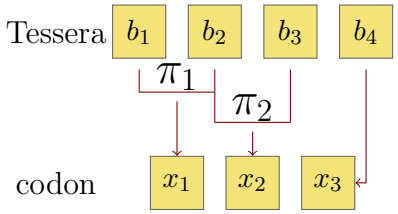
3.3.2 Properties of maximal circular Tessera codes

In this section, properties of maximal circular Tessera codes are listed. All properties in this section are based on the construction of Tessera codes in section 3.3.1.

Proposition 3.3.5. *Let us assume that $\mathcal{X}$ is a circular Tessera code. The longest path in $\mathcal{G}(\mathcal{X})$ can be 3 at most (see Proposition 3.2.1). To refine this suggestion, there can be no more than four of these paths with a length of 3.*

Proof. The longest path in a tournament with four vertices is 3 at most. A circular Tessera code has four disjoint components in the $\mathcal{C}_2(\mathcal{X})$. Each of these components can be extended to a tournament and can therefore only have a path of length 3. The longest path in the $\mathcal{C}_1(\mathcal{X})$ is 2 at most and can therefore be ignored. This proves the proposal. □

The next observation is based on the trinucleotide codes resolving from the mapping function $cod(\cdot)$ [40] for passing from the Tessera code back to the (mitochondrial) genetic code. Let us recall $cod(\cdot)$ by displaying it in Figure 3.3.



Copy of Figure 3.3 Schematic representation of the mapping function $cod(\cdot)$ from a Tessera $b_1b_2b_3b_4$ onto the codon $x_1x_2x_3$.

Let us set the assignment of the letters in the genetic alphabet to each of the transformations from $\mathcal{V}$: I $\leftrightarrow$ A, SW $\leftrightarrow$ T, KM $\leftrightarrow$ C and YR $\leftrightarrow$ G.

Observation 3.3.1. *Let $\beta \in \mathcal{V}$ and $\mathcal{TE}_{beta} \subset \mathcal{TE}$ be one of the subsets defined in Definition 23. By applying the mapping function $cod(\cdot)$ to any circular code $\mathcal{X}_\beta \subset \mathcal{TE}_\beta$ we obtain a trinucleotide code $\mathcal{X} = cod(\mathcal{X}_\beta)$. The obtained codes of all possible $\mathcal{X}_\beta$ are comma-free codes. Additionally, each trinucleotide code $\mathcal{X} = cod(\mathcal{X}_\beta)$ is also comma-free for any circular j-permutation of $\mathcal{X}$.*

This observation is used in section 6.1 as one element for a hypothesis of the evolution.

Self-complementary

This paragraph presents the characteristics concerning the class of self-complementarity circular Tessera codes.

Proposition 3.3.6. *Let $\mathcal{X} \subset \mathcal{TE}$ be a self-complementary circular Tessera code. Then $\mathcal{X} \cap \mathcal{TE}_{SW} = \emptyset$.*

Proof. In short: $\forall w \in \mathcal{TE}_{SW} \rightarrow \overleftarrow{c(w)} = \alpha_2(w)$. This fact is made evident by the following word construction:

$$\gamma \in \mathcal{V} \setminus I, b_1 \in \{A, T, G, C\}$$

$$w = b_1 SW(b_1)\gamma(b_1)(\gamma \circ SW)(b_1) = b_1b_2b_3b_4 \rightarrow w \in \mathcal{TE}_{SW}$$

$$\overleftarrow{c(w)} = SW(b_4b_3b_2b_1) = \gamma(b_1)(\gamma \circ SW)(b_1)b_1SW(b_1) = \alpha_2(w)$$

□

The proof of Proposal 3.3.6 demonstrates that a self-complementary circular Tessera code $\mathcal{X}$ consists only of three fragments:

$$\mathcal{X} = \mathcal{X}_I \cup \mathcal{X}_{YR} \cup \mathcal{X}_{KM}$$

$$\text{where } |\mathcal{X}_I| \leq 6, |\mathcal{X}_{YR}| \leq 6, |\mathcal{X}_{KM}| \leq 6$$

Proposition 3.3.7. *Let $\mathcal{X} \subset \mathcal{TE}$ be a self-complementary circular Tessera code and $\mathcal{X}_\beta$ a fragment of $\mathcal{X}$. Then, for all $w \in \alpha_1(\mathcal{X}_\beta) \cap \mathcal{TE}_{SW}$, it follows that $\overleftarrow{c(w)} = w$.*

Proof. Let $\mathcal{X} \subset \mathcal{TE}$ be a self-complementary circular Tessera code and $w = b_1b_2b_3b_4 \in \mathcal{X}$. Further, assume that $\beta, \gamma \in \mathcal{V}$, so that $SW = \gamma \circ \beta$ with $b_1\beta(b_1) = b_1b_2 = \gamma(b_3b_4)$(see Figures 3.9-3.12, where these words are represented by the blue edges). For all of them, it can be said that $\overleftarrow{c(w)} = w$. This is proven in detail as follows:

$$\beta \in \mathcal{V} \setminus I, \gamma = SW \circ \beta, b_1 \in \{A, T, G, C\}$$

$$w = b_1\beta(b_1)\gamma(b_1)SW(b_1) = b_1b_2b_3b_4 \rightarrow \alpha_1(w), \alpha_3(w) \in \mathcal{TE}_{SW}$$

$$\overleftarrow{c(w)} = (SW \circ SW)(b_1)(SW \circ \gamma)(b_1)(SW \circ \beta)(b_1)SW(b_1) = b_1b_2b_3b_4$$

This proves that for all $w \in \alpha_1(\mathcal{X}_\beta) \cup \mathcal{TE}_{SW}$, it follows that $\overleftarrow{c(w)} = w$. □

Proposition 3.3.7 shows that self-complementary circular Tessera codes can have an odd size, which is impossible for self-complementary circular trinucleotide codes.

Proposition 3.3.8. *A single tournament representing one of the equivalence classes (table 3.3) is self-complementary only if the numbers 1, 2, 3 and 4 (paragraph Construct a Tournament) are assigned to the dinucleotides in such a way that 1 is complementary to 4 and 2 is complementary to 3.*

Example 24.1. *As an example, we use the class $\mathfrak{B}_{KM}$. In this class, one possible self-complementary assignment would be: $1 \rightarrow CT$, $4 \rightarrow AG$, $2 \rightarrow TC$ and $3 \rightarrow GA$. The code $\mathcal{X}_{KM} = \{CTAG, CTTC, CTGA, TCAG, TCGA, GAAG\}$ represented by the tournament is self-complementary.*

Proof. This can be explained by using the graph property of self-complementary codes. The property states that each vertex v in the graph of self-complementary codes must have the same incoming degree as the outgoing degree of the complementary vertex $c(\overleftarrow{v})$ (see Proposition 2.1.2) and vice versa:

$$d^+(c(\overleftarrow{v})) = d^-(v)$$

In a tournament, this can only be true for the vertices assigned to 1 and 4 as well as for the vertices assigned to 2 and 3.

□

The following is a property of $\mathcal{G}(\mathcal{X})$, which was discovered in the study of maximum circular codes of codons (RNA triplets) and has already been depicted in section 2.1.5. Suppose $Y \subset \Sigma^3$ is such a maximum circular code. These conditions address the vertices $V(Y) \in \mathcal{G}(Y)$ associated with Y. Thus, condition (1) and (2) of Theorem 3.3.1 apply to all vertices V but only if Y is self-complementary.

Theorem 3.3.1. *This theorem applies the same conditions as in Theorem 2.1.5 to circular Tessera codes. Let $\mathcal{X} \subset \mathcal{TE}$ be a circular Tessera code. Then $\mathcal{X}$ is self-complementary if and only if*

1. $V(\mathcal{X}) = \overleftarrow{c(V(\mathcal{X}))}$
2. $d^+(v) = d^-(\overleftarrow{c(v)})$ *for all vertices* $v \in VV(\mathcal{X})$

Proof. The obligatory existence of condition (1) and (2) of Theorem 3.3.1 in a graph representing a self-complementary circular Tessera code can be proven in the two components $\mathcal{C}_1$ and $\mathcal{C}_2$. For Tessera codes, the conditions fit even better because they are not bound to the size of the code.

$\mathcal{C}_1(\mathcal{X})$ Let us assume that $\mathcal{X} \subset \mathcal{TE}$ is a non-self-complementary circular Tessera code, and $\mathcal{G}(\mathcal{X})$ fulfills the conditions. Any word $b_1b_2b_3b_4 \in \mathcal{X}$ appears in $\mathcal{C}_1$ as $b_1 \to b_2b_3b_4$. In order to satisfy the conditions, $c(b_4b_3b_2) \to b_5$ must be in $\mathcal{C}_1$. Since each word in $\mathcal{X}$ is a Tessera, and therefore $c(b_4b_3b_2)b_5$ is a Tessera, it follows that $c(b_1) = b_5$. These results show that the conditions are only true if $\mathcal{X}$ is a self-complementary code. Thus, this is a contradiction.

$\mathcal{C}_2(\mathcal{X})$ These conditions must also apply to any $\mathcal{C}_2(\mathcal{X})$ that represents a circular self-complementary Tessera code $\mathcal{X} \subset \mathcal{TE}$. The method to prove this is the Pigeonhole principle. The proof shows that condition (1) and (2) of Theorem 3.3.1 must be true for each tournament, which can then easily be extended to the entire graph. Consequently, it is only necessary to prove this for a $\mathcal{C}_2$ of a single fragment of a circular Tessera code.

Assume that $\mathcal{X}_\gamma \subset \mathcal{TE}_\gamma$ is a non-self-complementary circular Tessera code, and $\mathcal{C}_2(\mathcal{X}_\gamma)$ satisfies condition (1) and (2) of Theorem 3.3.1. If $d^+(v) = d^-(\overleftarrow{c(v)})$, it follows that $\mathcal{C}_2(\mathcal{TE}_\gamma)$ cannot be a tournament (see Proposition 3.3.8). Thus, $|\mathcal{X}_\gamma| < 6$. Since all the words in the form of $b_1b_2\overleftarrow{c(b_1b_2)}$ are self-complementary, none of them have any influence on this proof and can be ignored. Even though $\mathcal{X}_{SW}$ does not contain a word of this form, it can be said that $|\mathcal{X}_\gamma|$ is even. Therefore, $|\mathcal{X}_\gamma| \in \{2, 4\}$. Considering all of the above, the $\mathcal{C}_2(\mathcal{X}_\gamma)$ must be regular. In the case of $|\mathcal{X}_\gamma| = 4$, it must be 2-regular. In the case of $|\mathcal{X}_\gamma| = 2$, it must be 1-regular. The number

of options of a 1-regular graph with two edges requires that the represented code is self-complementary and therefore contradictory.

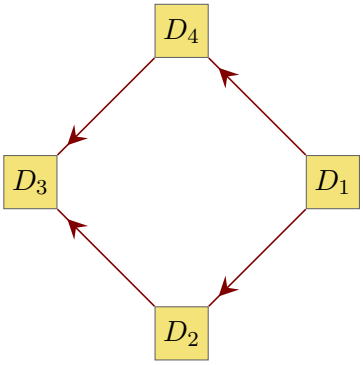


Figure 3.8: 2-regular $\mathcal{C}_2(\mathcal{X}_\gamma)$. The only 2-regular acyclic graph with four vertices which can satisfy conditions (1) and (2) of Theorem 3.3.1 where no complementary vertices are connected.

A directed acyclic 2-regular graph as in Figure 3.8 only satisfies condition (2) of Theorem 3.3.1: $d^+(v) = d^-(\overleftarrow{c(v)})$ if $D_2 = \overleftarrow{c(D_4)}$ and $D_1 = \overleftarrow{c(D_3)}$. Thus, this must also represent a self-complementary code and is therefore a contradiction. This proves that $\mathcal{C}_2(\mathcal{TE}_\gamma)$ is either non-self-complementary or satisfies condition (1) and (2) of Theorem 3.3.1.

□

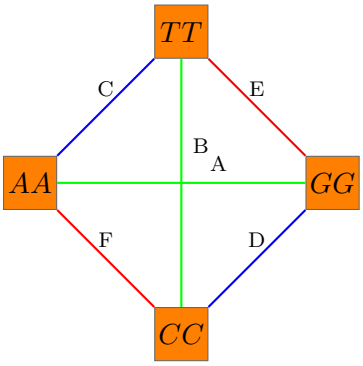


Figure 3.9: The $\mathfrak{B}_I$

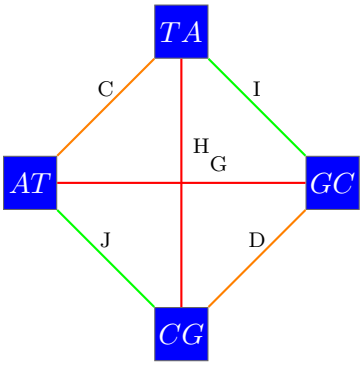


Figure 3.10: The $\mathfrak{B}_{SW}$

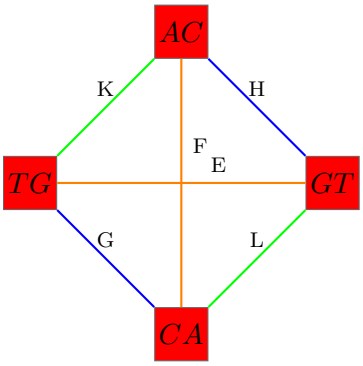


Figure 3.11: The $\mathfrak{B}_{YR}$

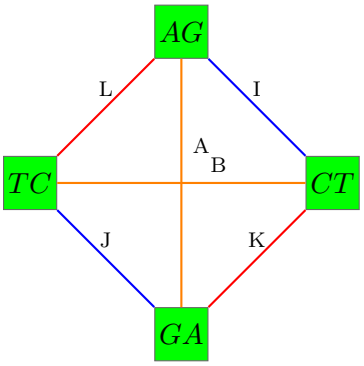


Figure 3.12: The $\mathfrak{B}_{KM}$

Figure 3.9-3.12: The four undirected graphs represent the four equivalence classes. Each edge represents two Tessera words. The edge labels (letters: A-L) connect the circular equivalent words. For better orientation, the color of the edges corresponds with the colors of the nodes of the circular equivalent words.

3.4 Refined construction of circular and comma-free Tessera codes

The number $2^{12} \times (4!)^4 = 1358954496$ presented in section 3.3.1 was a first attempt to calculate the the exact number of maximum circular Tessera codes. Even though the number includes all maximum circular Tessera codes, as mentioned above, some codes appear more than once. This section outlines an algorithm that uses characteristics of codes to determine how often each maximum circular Tessera code occurs in this number. The result of this algorithm allows to calculate the exact number of different circular Tessera codes of any size from one to twelve.

The number 1358954496 results from the idea that the $(4!)^4$ codes of size 24 are each converted to 2^{12} circular codes. Suppose that $\mathcal{X}'$ is a representative of these

$(4!)^4$ constructed codes. 2^{12} indicates the twelve circular equivalence relationships within each code. As mentioned above, there are 2^{12} different ways to select a w from each circular pair of relationships in $\mathcal{X}'$. Additionally, it is obvious that all 2^{12} circular codes that resolve from this method are different. However, suppose that $\mathcal{Y}'$ is another code from the $(4!)^4$ Tessera codes. $\mathcal{Y}'$ could be used to construct circular codes that are similar to the circular codes constructed from $\mathcal{X}'$. In the worst case, $\mathcal{X}'$ and $\mathcal{Y}'$ differ in one word only. In such a case, 2^{11} of the codes constructed from $\mathcal{X}'$ and $\mathcal{Y}'$ are the same.

To refine the construction so that each code is constructed only once, the new algorithm is based on two categories of features. One of these is called the *fragment distribution*, the other the *circular permutation distribution*. The *fragment distribution* refers to the different options available to distribute the Tesserae in a code to the code fragments introduced in Definition 24. The *circular permutation distribution* refers to the 4! construction of the four tournaments (see paragraph Construct a Tournament).

3.4.1 Fragment distribution

The fragment distribution refers to the distributions of the Tesserae in a code $\mathcal{X} \subset \mathcal{TE}$ into the four code fragments $\mathcal{X}_I$, $\mathcal{X}_{SW}$, $\mathcal{X}_{YR}$ and $\mathcal{X}_{KM}$. The originally constructed codes $\mathcal{X}'$ in section 3.3.1 have six Tesserae in each fragment:

$$|\mathcal{X}_I| = |\mathcal{X}_{SW}| = |\mathcal{X}_{YR}| = |\mathcal{X}_{KM}| = 6 \text{ for all } (4!)^4 \text{ codes } \mathcal{X}'$$

The circular code $\mathcal{X}$ can only have twelve Tesserae at the most. All possible distributions are listed in Table 3.4

Table 3.4: A list of all possible distributions of circular codes into the code fragments $\mathcal{X}_I$, $\mathcal{X}_{SW}$, $\mathcal{X}_{YR}$ and $\mathcal{X}_{KM}$. Each section in the table shows all fragments of codes of a certain code size. The distributions listed range from a code of size four to size twelve, which is the maximum.

code size	Distributions				code size	Distributions				code size	Distributions			
6	6	0	0	0	5	5	0	0	0	4	4	0	0	0
	5	1	0	0		4	1	0	0		3	1	0	0
	4	2	0	0		3	2	0	0		2	2	0	0
	4	1	1	0		3	1	1	0		2	1	1	0
	3	3	0	0		2	2	1	0		1	1	1	1
	3	2	1	0		2	1	1	1	9	6	3	0	0
	3	1	1	1	8	6	2	0	0		6	2	1	0
	2	2	2	0		6	1	1	0		6	1	1	1
	2	2	1	1		5	3	0	0		5	4	0	0
7	6	1	0	0		5	2	1	0		5	3	1	0
	5	2	0	0		5	1	1	1		5	2	2	0
	5	1	1	0		4	4	0	0		5	2	1	1
	4	3	0	0		4	3	1	0		4	4	1	0
	4	2	1	0		4	2	2	0		4	3	2	0
	4	1	1	1		4	2	1	1		4	3	1	1
	3	3	1	0		3	3	2	0		4	2	2	1
	3	2	2	0		3	3	1	1		3	3	3	0
	3	2	1	1		3	2	2	1		3	3	2	1
	2	2	2	1		2	2	2	2		3	2	2	2
10	6	4	0	0	11	6	4	1	0	12	6	4	2	0
	6	3	1	0		6	3	2	0		6	4	1	1
	6	2	2	0		6	3	1	1		6	3	3	0
	6	2	1	1		6	2	2	1		6	3	2	1
	5	5	0	0		5	5	1	0		6	2	2	2
	5	4	1	0		5	4	2	0		5	5	2	0
	5	3	2	0		5	4	1	1		5	5	1	1
	5	3	1	1		5	3	3	0		5	4	3	0
	5	2	2	1		5	3	2	1		5	4	2	1
	4	4	2	0		5	2	2	2		5	3	3	1
	4	4	1	1		4	4	3	0		5	3	2	2
	4	3	3	0		4	4	2	1		4	4	4	0
	4	3	2	1		4	3	3	1		4	4	3	1
	4	2	2	2		4	3	2	2		4	4	2	2
	3	3	3	1		3	3	3	2		4	3	3	2
	3	3	2	2							3	3	3	3

The distributions are bound by four rules. First, the highest number must be ≤ 6. Secondly, the sum must be equal to the code size. Thirdly, if one of the fragments in one of the distributions is 6, it follows that all other fragments are 4 at the most. Finally, if one of the fragments is 5, it follows that all other fragments are 5 or less. This is due to the circular relationships between the fragments shown in Proposition 3.3.3.

The following pseudocode algorithm provides all possible distributions for the circular Tessera codes with a code size called l. The values returned are in numerical order:

Pseudocode 2. ***Require:*** $12 \geq l \in \mathbb{N} \geq 1$

```
function DISTRIBUTIONS(l:int)
    distributions : list < list >← []
    upperBound : int ← min(10, l)
    for a ∈ 1 ... min(6, l) do
        for b ∈ ⌈(l − a)/3⌉ ... min(upperBound − a, a)) do
            for c ∈ ⌈(l − a − b)/2⌉ ... min(upperBound − a − b, b)) do
                distributions.add([a, b, c, l − a − b − c])
            end for
        end for
    end for
    return distributions
end function
```

This paragraph explains the algorithm line by line. The explanation starts with the assignment of *distributions* and ends with line 6: $distributions.add([a, b, c, l - a - b - c])$.

1. allocates an empty agile list of lists to "*distributions*".

2. allocates an integer to "*upperBound*. The value is the minimum of 10 and l. The number 10 results from the fact that all fragments are in two circular relations to each other. This means that in any combination of $\mathcal{X}_1, \mathcal{X}_2 \in \{\mathcal{X}_I, \mathcal{X}_{SW}, \mathcal{X}_{YR}, \mathcal{X}_{KM}\}$, it follows that $|\mathcal{X}_1 \cup \mathcal{X}_2| \leq 10$ if the code is circular.

3. a for loop from 1 to the minimum of 6 and l. The loop counter variable is called "a". The minimum value is 1, since the number 0 would not terminate. "a" is the first fragment and can therefore not be higher than 6.

4. a for loop from $\lceil (l - a)/3 \rceil$ to the minimum of $upperBound - a$ and a. The loop-counter variable is the second fragment and is denoted as "b". The minimum value is $\lceil (l - a)/3 \rceil$. Consequently, the sum of all fragments must be l. It follows that $b \leq l - a$. However, the rest can be divided into three

fragments, which gives the lowest possible number for "$b \leq \lceil (l-a)/3 \rceil$" . The upper bound of the for loop $upperBound - a$ or a is derived from the definition of "*upperBound*" (see line 2).

5. a for loop from $\lceil (l-a-b)/2 \rceil$ to the minimum of $upperBound - a - b$ and b. The loop counter variable is the third fragment and is called "c". The minimum value is $\lceil (l-a-b)/2 \rceil$. Consequently, the sum of all fragments must be l. $c \leq l - a - b$ follows. However, the rest can be divided into two fragments, which gives the lowest possible number for "$c \leq \lceil (l-a-b)/2 \rceil$". The upper limit of the for-loop $upperBound - a - b$ or b results from the definition of "*upperBound*". (see line 2).

6. assembles the distribution and adds it to "*distributions*". The first fragment is "a", second fragment "b", third fragment "c", and the last fragment, which is the rest, is $l - a - b - c$.

3.4.2 Circular permutation distribution (CPD)

The CPD refers to the selection of Tesserae within each fragment of a code. In Proposal 3.3.3, it is revealed that a circular fragment can contain six of twelve possible Tesserae. It also presents evidence that these six can be classified by their circular 1-permutations and circular 3-permutations into three classes of two. We define the CPD as the different ways to distribute the Tesserae in a circular fragment to the classes, i.e. each class includes the two Tesserae that have their circular 1-permutations and circular 3-permutations in the same other fragment. In Definition 25, we define the classes in these fragments. The classes are labeled as A_i, B_i and C_i for a fragment $\mathcal{X}_i$.

Definition 25. *Suppose $\mathfrak{B}_1$, $\mathfrak{B}_2$, $\mathfrak{B}_3$ and $\mathfrak{B}_4$ are the four equivalence classes shown in Table 3.3:*

$$\{\mathfrak{B}_1, \mathfrak{B}_2, \mathfrak{B}_3, \mathfrak{B}_4\} \coloneqq \{\mathfrak{B}_I, \mathfrak{B}_{SW}, \mathfrak{B}_{KW}, \mathfrak{B}_{YR}\}$$

and $\mathcal{X}_1$, $\mathcal{X}_2$, $\mathcal{X}_3$ and $\mathcal{X}_4$ are the four fragments associated with the four equivalence classes. We use the index $1-4$ as an implied order. The exact mapping is irrelevant:

$$\mathcal{TE}_i \coloneqq \{D_1 D_2 : D_1, D_2 \in \mathfrak{B}_i\} \text{ for all } i \in \{1,2,3,4\}$$

where D_1, D_2, D_3 and D_4 are denoted as the four dinucleotides in $\mathfrak{B}_i$:

$$\mathfrak{B}_i \coloneqq \{D_1, D_2, D_3, D_4\}$$

Let $D_1, D_2, D_3, D_4 \in \mathfrak{B}_1$ and $A_1 \cup B_1 \cup C_1 \coloneqq \mathcal{TE}_1$	
Class	Circular relationship
$A_1 = \{D_1D_2, D_3D_4, D_2D_1, D_4D_3\}$	$\alpha_1(A_1) = \alpha_3(A_1) \coloneqq \mathcal{TE}_2 \cap A_2$
$B_1 = \{D_1D_3, D_2D_4, D_3D_1, D_4D_2\}$	$\alpha_1(B_1) = \alpha_3(B_1) \coloneqq \mathcal{TE}_3 \cap A_3$
$C_1 = \{D_1D_4, D_2D_3, D_4D_1, D_3D_2\}$	$\alpha_1(C_1) = \alpha_3(C_1) \coloneqq \mathcal{TE}_4 \cap A_4$
Let $D_1, D_2, D_3, D_4 \in \mathfrak{B}_2$ and $A_2 \cup B_2 \cup C_2 \coloneqq \mathcal{TE}_2$	
Class	Circular relationship
$A_2 = \{D_1D_2, D_3D_4, D_2D_1, D_4D_3\}$	$\alpha_1(A_2) = \alpha_3(A_2) \coloneqq \mathcal{TE}_1 \cap A_1$
$B_2 = \{D_1D_3, D_2D_4, D_3D_1, D_4D_2\}$	$\alpha_1(B_2) = \alpha_3(B_2) \coloneqq \mathcal{TE}_3 \cap B_3$
$C_2 = \{D_1D_4, D_2D_3, D_4D_1, D_3D_2\}$	$\alpha_1(C_2) = \alpha_3(C_2) \coloneqq \mathcal{TE}_4 \cap B_4$
Let $D_1, D_2, D_3, D_4 \in \mathfrak{B}_3$ and $A_3 \cup B_3 \cup C_3 \coloneqq \mathcal{TE}_3$	
Class	Circular relationship
$A_3 = \{D_1D_2, D_3D_4, D_2D_1, D_4D_3\}$	$\alpha_1(A_3) = \alpha_3(A_3) \coloneqq \mathcal{TE}_1 \cap B_1$
$B_3 = \{D_1D_3, D_2D_4, D_3D_1, D_4D_2\}$	$\alpha_1(B_3) = \alpha_3(B_3) \coloneqq \mathcal{TE}_2 \cap B_2$
$C_3 = \{D_1D_4, D_2D_3, D_4D_1, D_3D_2\}$	$\alpha_1(C_3) = \alpha_3(C_3) \coloneqq \mathcal{TE}_4 \cap C_4$
Let $D_1, D_2, D_3, D_4 \in \mathfrak{B}_4$ and $A_4 \cup B_4 \cup C_4 \coloneqq \mathcal{TE}_4$	
Class	Circular relationship
$A_4 = \{D_1D_2, D_3D_4, D_2D_1, D_4D_3\}$	$\alpha_1(A_4) = \alpha_3(A_4) \coloneqq \mathcal{TE}_1 \cap C_1$
$B_4 = \{D_1D_3, D_2D_4, D_3D_1, D_4D_2\}$	$\alpha_1(B_4) = \alpha_3(B_4) \coloneqq \mathcal{TE}_2 \cap C_2$
$C_4 = \{D_1D_4, D_2D_3, D_4D_1, D_3D_2\}$	$\alpha_1(C_4) = \alpha_3(C_4) \coloneqq \mathcal{TE}_3 \cap C_3$

Table 3.5: The classification of the Tesserae in a fragment $\mathcal{X}_i$. Each set A_i, B_i, C_i can only contain a maximum of two out of four Tesserae for them to be circular.

As revealed in Table 3.5, each class in a fragment has a counterpart in another fragment e.g. the counterpart of $A_1 \in \mathcal{TE}_1$ is $A_2 \in \mathcal{TE}_2$ and vice versa. It is evident that A_1 contains four Tesserae originating form two circular permutation classes, and A_2 contains the other four Tesserae from to the same two circular permutation classes. Hence, if we assume that $\mathcal{X} \subset \mathcal{TE}$ is a circular Tessera code and $Z = \mathcal{X} \cap (A_1 \cup A_2)$, then $|Z| \leq 2$ must follow.

Let us explain the consequences of Definition 25 using an example without loss of generality. Let $\mathcal{X} \subset \mathcal{TE}$ be a circular Tessera code and $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3 \cup \mathcal{X}_4$ is the fragmented representation of $\mathcal{X}$. Assume that $w_1, w_2 \in \mathcal{X}_1 \cap A_1 \cup \mathcal{X}$. Considering that $\mathcal{X}_2$ is the fragment containing all 1-permutations and 3-permutations of the Tesserae in A_1, then, with respect to Table 3.5, it follows that $|\mathcal{X}_2| \leq 4$, more precisely, class $A_2 = \emptyset$ of the fragment $\mathcal{X}_2$. This occurrence can be used to construct a tabular representation of any circular Tessera code.

$\lvert\mathcal{X}_1\rvert$	$\lvert A_1\rvert$	$\lvert B_1\rvert$	$\lvert C_1\rvert$
$\lvert A_2\rvert$	$\lvert\mathcal{X}_2\rvert$	$\lvert B_2\rvert$	$\lvert C_2\rvert$
$\lvert A_3\rvert$	$\lvert B_3\rvert$	$\lvert\mathcal{X}_3\rvert$	$\lvert C_3\rvert$
$\lvert A_4\rvert$	$\lvert B_4\rvert$	$\lvert C_4\rvert$	$\lvert\mathcal{X}_4\rvert$

Table 3.6: This table $\mathcal{T}(\mathcal{X})$ represents any circular Tessera code $\mathcal{X} \subset \mathcal{TE}$. $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3 \cup \mathcal{X}_4$ are the four fragments. A_i, B_i, C_i are the three classes of each fragment $\mathcal{X}_i$ in regard to Table 3.5. Let $\mathcal{T}_{r,c}$ be the value in the table at row r and column c. Then, the Tesserae represented at a position $\mathcal{T}_{r,c}$ are the circular relatives of the Tesserae represented in $\mathcal{T}_{c,r}$. Hence, $0 \leq \mathcal{T}_{r,c} + \mathcal{T}_{c,r} \leq 2$, with $r, c \in \{1, 2, 3, 4\}$ and $r \neq c$

The tables are called representing tables of a Tessera code $\mathcal{T}(\mathcal{TE})$. From now on $\mathcal{X}_1$ to $\mathcal{X}_4$ are used as placeholders for the four fragments $\mathcal{X}_I$, $\mathcal{X}_{SW}$, $\mathcal{X}_{KM}$ and $\mathcal{X}_{YR}$. In relation to a table $\mathcal{T}(\mathcal{X})$, $\mathcal{X}_1$ to $\mathcal{X}_4$ are also used as row references.

Definition 26. *Any circular Tessera code can be represented in such a table. A table $\mathcal{T}$ representing a circular Tessera code $\mathcal{X}$ must be bound to the following conditions:*

1. $|\mathcal{X}_i| \leq 6$
2. $|\mathcal{X}_i| = |A_i| + |B_i| + |C_i|$
3. $0 \leq |A_i| \leq 2$ *&* $0 \leq |B_i| \leq 2$ *&* $0 \leq |C_i| \leq 2$
4. $R = \begin{bmatrix} |A_2| & |A_3| & |A_4| \\ |A_1| & |B_3| & |B_4| \\ |B_1| & |B_2| & |C_4| \\ |C_1| & |C_2| & |C_3| \end{bmatrix} + \begin{bmatrix} |A_1| & |B_1| & |C_1| \\ |A_2| & |B_2| & |C_2| \\ |A_3| & |B_3| & |C_3| \\ |A_4| & |B_4| & |C_4| \end{bmatrix}$ *so that* $\forall \lambda \in R : 0 \leq \lambda \leq 2$

Each table which is built according to the pattern of Table 3.6 represents at least one circular Tessera code. This applies to all tables, but only as long as the listed conditions in Definition 26 are satisfied.[4]

Example 26.1. *Let us assume that* $\mathcal{X} = \{AACC, AAGG, AATT, GGCC, GGTT, CCTT, TAGC, TACG, CGAT, ATGC, TGAC, GTCA\}$

$$\begin{aligned} \mathcal{X}_I &= \{AACC, AAGG, AATT, GGCC, GGTT, CCTT\} \\ \mathcal{X}_{SW} &= \{TAGC, TACG, CGAT, ATGC\} \\ \mathcal{X}_{YR} &= \{TGAC, GTCA\} \\ \mathcal{X}_{KM} &= \emptyset \end{aligned}$$

[4]The algorithm that compiles all possible CPD for all fragment distributions is too extensive to be explained here. The algorithm is implemented in *GCATR* and can be found in the Github reposetory: `https://github.com/StarmanMartin/GCATR`

The fragment distribution in this example is chosen as: $\mathcal{X}_1 = \mathcal{X}_I$, $\mathcal{X}_2 = \mathcal{X}_{SW}$, $\mathcal{X}_3 = \mathcal{X}_{YR}$ *and* $\mathcal{X}_4 = \mathcal{X}_{KM}$. *Hence, the values in the tables are:* $6 = |\mathcal{X}_1|$, $4 = |\mathcal{X}_2|$,$2 = |\mathcal{X}_3|$ *and* $0 = |\mathcal{X}_4|$

6	2	2	2
0	4	2	2
0	0	2	2
0	0	0	0

Table 3.7: Among others, this table $\mathcal{T}(\mathcal{X})$ represents the following code $\mathcal{X} = \{AACC, AAGG, AATT, GGCC, GGTT, CCTT, TAGC, TACG, CGAT, ATGC, TGAC, GTCA\}$. $\mathcal{X}_1 = \mathcal{X}_I$, $\mathcal{X}_2 = \mathcal{X}_{SW}$, $\mathcal{X}_3 = \mathcal{X}_{YR}$ and $\mathcal{X}_4 = \mathcal{X}_{KM}$.

3.4.3 The exact number of circular Tessera codes

An approach derived from combinatorics is used to calculate the exact number of codes. We first introduce the basics of probability theory on which the calculation of the exact number of Tessera codes is based. Each table $\mathcal{T}$ can represent no more than $(4!)^5$ codes. This number results from the 4! different ways to allocate the four fragments $\mathcal{X}_I$, $\mathcal{X}_{SW}$, $\mathcal{X}_{KM}$ to the table rows $\mathcal{X}_1 - \mathcal{X}_4$ and the 4! different structures of the four tournaments.

Definition 27. *Let us designate every single combination of fragment allocation and tournament structure as one of* $(4!)^5$ *parameter sets* $\theta \in \Theta$ *of a table. Each element in the parameter sets* θ *is a tuple*

$$\theta \coloneqq (\theta_o, \theta_i)$$

where θ_o *stands for the fragment allocation, and* θ_i *is denoted as one of the* $(4!)^4$ *tournament structures.*

Definition 27 introduces the *parameter tuples* $\theta \coloneqq (\theta_o, \theta_i)$. To reconstruct a code from a table $\mathcal{T}$ one needs to apply the correct θ to the table.

$$\mathcal{X} = \theta \cdot \mathcal{T}$$

Definition 28. *Let the tuple* (Ω, Pr) *be a probability space.* Ω *is a set called sample set, and Pr is a function which we call probability function. The function is defined so that* $0 \leq Pr(\mathcal{X}) \leq 1$ *for all* $\mathcal{X} \in \Omega$ *and* $\sum_{\mathcal{X} \in \Omega} Pr(\mathcal{X}) = 1$

In this case, we refer to Ω *as a set of all circular Tessera codes* $\mathcal{X}$ *represented by a table. Thus,* $Pr(\mathcal{X}) = \frac{1}{|\Omega|}$ *is valid.* $Pr(\mathcal{X})$ *is the probability that an arbitrarily chosen parameter tuple* $\theta \in \Theta$ *applied on a table constructs the code* $\mathcal{X}$.

The probability $Pr(\mathcal{X})$ for a code can be separated into the so-called *inner probability* and the so-called *outer probability*. The *inner probability* refers to the CPD and the $(4!)^4$ different combinations of the tournaments. The *outer probability* refers to the fragment distributions. The first part of this section explains the *outer probability* $Pr_o(\mathcal{X})$, which depends on θ_o. The second part of this section refers to the *inner probability* denoted as $Pr_i(\mathcal{X})$, which depends on θ_i.

The *outer probability* is a number $Pr_o(\mathcal{X})$ which indicates how likely it is to obtain a code $\mathcal{X}$ when we apply an arbitrary parameter θ_o (ignoring θ_i) to a table $\mathcal{T}(\mathcal{X})$ which represents the code $\mathcal{X}$. Hence, the focus is on a certain assignment of the rows designated as $\mathcal{X}_1$ to $\mathcal{X}_4$ to the four different code fragments.

$$\{\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \mathcal{X}_4\} \rightarrow \{\mathcal{X}_I, \mathcal{X}_{SW}, \mathcal{X}_{KW}, \mathcal{X}_{YR}\}$$

There are 4! different ways to assign $\mathcal{X}_1$ to $\mathcal{X}_4$ to the four fragments. In the case of example 26.1, the assignment is $\mathcal{X}_1 = \mathcal{X}_I$, $\mathcal{X}_2 = \mathcal{X}_{SW}$, $\mathcal{X}_3 = \mathcal{X}_{YR}$ and $\mathcal{X}_4 = \mathcal{X}_{KM}$. In this particular case, the probability is $Pr_o(\mathcal{X}) = \frac{1}{4!} \times 1$. This is due to the fact that two different assignments of the fragments cannot represent the same code. Therefore, the distribution in example 26.1 has 4! codes (times the result of $Pr_i(\mathcal{X})$). In the theoretical case that two rows of a table can be exchanged without affecting the table, $Pr_o(\mathcal{X}) = \frac{1}{4!} \times 2$.

In order to generalize this assumption, let us consider the 4! mappings from $\mathcal{X}_1$ to $\mathcal{X}_4$ as row transformations of a table $\mathcal{T}$ associated with the code $\mathcal{X}$, which can be seen in Table 3.8. Suppose G is a transformation group with all possible row transformations of cardinality 4!:

All actions in G are defined on a set of all tables M_l. M_l is a set of all tables $\mathcal{T}$, which represent all codes of one code size l so that:

$$\varphi \colon G \times M_l \rightarrow M_l$$

This function satisfies all the action group axioms. A stabilizer subgroup G_s of G with respect to $\mathcal{T}$ is defined as:

$$G_s = \{g \in G \mid g \cdot \mathcal{T} = \mathcal{T}\}.$$

Let us assume that G_s is a stabilizer subgroup with respect to $\mathcal{T}$. Generally, it can be said that $Pr_o(\mathcal{X})$ is equal to $\frac{1}{4!}$ times the cardinality of G_s.

$\lvert\mathcal{X}_1\rvert$	$\lvert A_1\rvert$	$\lvert B_1\rvert$	$\lvert C_1\rvert$
$\lvert A_2\rvert$	$\lvert\mathcal{X}_2\rvert$	$\lvert B_2\rvert$	$\lvert C_2\rvert$
$\lvert A_3\rvert$	$\lvert B_3\rvert$	$\lvert\mathcal{X}_3\rvert$	$\lvert C_3\rvert$
$\lvert A_4\rvert$	$\lvert B_4\rvert$	$\lvert C_4\rvert$	$\lvert\mathcal{X}_4\rvert$

$$(X_1, X_2, X_3, X_4) \longrightarrow (X_3, X_2, X_1, X_4)$$

$\lvert\mathcal{X}_3\rvert$	$\lvert A_3\rvert$	$\lvert B_3\rvert$	$\lvert C_3\rvert$
$\lvert B_2\rvert$	$\lvert\mathcal{X}_2\rvert$	$\lvert A_2\rvert$	$\lvert C_2\rvert$
$\lvert A_1\rvert$	$\lvert B_1\rvert$	$\lvert\mathcal{X}_1\rvert$	$\lvert C_1\rvert$
$\lvert C_4\rvert$	$\lvert B_4\rvert$	$\lvert A_4\rvert$	$\lvert\mathcal{X}_4\rvert$

Table 3.8: The tables show an example of a row transformation. The transformation $g : (X_1, X_2, X_3, X_4) \rightarrow (X_3, X_2, X_1, X_4) \in G$ swaps rows 1 and 3 of a table $\mathcal{T}$. The transformation g is one of 4! transformations into G.

In conclusion, there are 4! different θ_o to chose from. Hence, $Pr_o(\mathcal{X}) \times 4!$ of all θ_o construct the same code when applied to a table. Since the probability of each code represented by the same table is the same $\Pr(\mathcal{X}) = \frac{1}{|\Omega|}$, it follows that a table $\mathcal{T}$ can construct $Pr_o(\mathcal{X})^{-1}$ different codes .

List II.2 in the Appendix depicts all $\mathcal{T}$ for all codes with the maximum length. The probability $Pr_o(\mathcal{X})$ is fixed for any table and can be used to calculate the number of comma-free codes or the codes of a certain path length in the 2-component $\mathcal{C}_2$ of a graph associated with a code. In all these calculations, the value Pr_o is bound to a $\mathcal{T}$ and remains the same.

The *inner probability* is a number $Pr_i(\mathcal{X})$ which indicates how likely it is to obtain a code $\mathcal{X}$ when we apply an arbitrary parameter θ_i (with a preselected and fixed θ_o in the tuples $\theta = (\theta_o, \theta_i)$) to a table $\mathcal{T}(\mathcal{X})$ which represents the code $\mathcal{X}$. The fragment distribution θ_o is considered as fixed for this calculation. Each 2-component assigned to a code fragment $\mathcal{X}_j$ can be extended in 4! different tournaments. The total number of Pr_i is therefore be considered to be a product of the probability of each row: $Pr_i(\mathcal{X}) \coloneqq Pr^*_{i(1)}(\mathcal{X}) \times Pr^*_{i(2)}(\mathcal{X}) \times Pr^*_{i(3)}(\mathcal{X}) \times Pr^*_{i(4)}(\mathcal{X})$. Consequently, there are 4! different θ_i per row to chose from. Thus, $Pr_i(\mathcal{X}) \times (4!)^4$ of all θ_i construct the same code when applied to a table. $Pr_i(\mathcal{X})^{-1} \times Pr_o(\mathcal{X})^{-1}$ different codes are represented by the table.

However, to simplify the calculation, $Pr_i(\mathcal{X})$ can be divided into two parts so that $Pr_i = Pr_{fac} \times Pr_{ri}$. Pr_{fac} is a value that depends on the interaction of all rows, wheras Pr_{ri} refers almost only to the single row.

Definition 29.

$$Pr_i(\mathcal{X}) \coloneqq Pr_{fac}(\mathcal{X}) \times Pr_{ri}(\mathcal{X})$$

with

$$Pr_{ri}(\mathcal{X}) \coloneqq Pr_{i(1)}(\mathcal{X}) \times Pr_{i(2)}(\mathcal{X}) \times Pr_{i(3)}(\mathcal{X}) \times Pr_{i(4)}(\mathcal{X})$$

and

$$Pr_{fac}(\mathcal{X}) \coloneqq \left(\prod_{r=1}^{r\leq 4} \prod_{c=1}^{c\leq 4} \begin{cases} \mathcal{T}_{r,c} mod(2) \times 2^{\frac{1}{1+\mathcal{T}_{c,r}}} & if\ r \neq c \\ 1 & if\ r = c \end{cases} \right)^{-1}$$

Pr_{fac} is a factor that has been isolated to simplify the calculation of Pr_{ri}. Each row probability $Pr_{i(r)}$ is defined so that the fact shown in Figure 3.13, which shows that every 1 in $\mathcal{T}$ stands for four Tesserae that have an influence on the number of possible Tesserae in another row, can be ignored. Instead, each 1 represents only one of two Tesserae. Therefore, each row can be treated separately. To compensate for this, Pr_{fac} is the factor which is 1 divided by 2 powers to the number of 1's. However, each pair of 1's in $\mathcal{T}_{r,c} = \mathcal{T}_{c,r}$ appears only once in the exponent.

Figure 3.13: Let us assume that $w_1, w_2, w_3, w_4 \in A_1$ (green vertices) and $w_5, w_6, w_7, w_8 \in A_2$ (red vertices) where $\mathcal{T}_{c,r} = |A_1|$ and $\mathcal{T}_{r,c} = |A_2|$. These two graphs simply depict the relations of the circular equivalence classes within a representing table $\mathcal{T}$. The graph shows that if $\mathcal{T}_{c,r} = \mathcal{T}_{r,c} = 1$, only one of the Tesserae in Figure A and one in Figure B can be in a code if the represented Tessera code is circular.

The complex Definition 29 of Pr_{fac} would be unnecessary for tables which represent only maximal circular Tessera codes of size twelve. Because in a $\mathcal{T}(\mathcal{X})$, which represents a maximal circular code $\mathcal{X}$, it follows that if $\mathcal{T}_{r,c} = 1$, then $\mathcal{T}_{c,r} = 1$. However, if the code is not maximal, this condition need not be true.

$Pr_{ri}(\mathcal{X})$ is the product of the probabilities that an arbitrarily θ_i applied to each row in $\mathcal{T}(\mathcal{X})$ will reconstruct a fragment $\mathcal{X}_j$ in $\mathcal{X}$. Let $\mathcal{X} \subset \mathcal{TE}$ be a circular Tessera code and $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3 \cup \mathcal{X}_4 \coloneqq \mathcal{X}$ its fragmented representation. Then:

$$Pr_{i(j)}(\mathcal{X}_j) = Pr_{i(j)}(\mathcal{X}) \text{ for } j \in \{1, 2, 3, 4\}$$

and

$$Pr_{ri}(\mathcal{X}) = \prod_{j=1}^{j \leq 4} Pr_{i(j)}(\mathcal{X}_j)$$

There are ten different patterns for the CPD (see Section 3.4.2). Since the introduction of Pr_{fac}, the order of the values does not matter anymore. Therefore, each line can be reduced to the values of $|A|$, $|B|$ and $|C|$. This is called a row

pattern and is written as the three values $|A|$, $|B|$ and $|C|$. For simplicity, they are written in a numerically ordered string. In Definition 30, we define all values for all row patterns. The subsequent section reveals the origin of these numbers.

Definition 30. *Let $|A||B||C|$ be a row pattern of a table $\mathcal{T}$ in row $j \in \{1,2,3,4\}$ and $\mathcal{X}_j$ the fragment represented in this row. Then the following list defines the results of $Pr_{i(j)}(\mathcal{X}_j)$:*

- *Row pattern 222* $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq \frac{1}{4!}$
- *Row pattern 122* $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq \frac{4}{4!\times 3} = \frac{1}{18}$
- *Row pattern 022* $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq \frac{12}{4!\times 7} = \frac{1}{14}$
- *Row pattern 112* $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq \frac{2}{4!} = \frac{1}{12}$
- *Row pattern 012* $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq \frac{3}{4!} = \frac{1}{8}$
- *Row pattern 111* $\rightarrow$ *Special case (see Table 3.9):*
  If $\mathcal{T}$ contains:
  $\leq$ 2 rows of the pattern 111 $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq 7^{-1}$
  3 rows of the pattern 111 $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq \left(\sqrt[3]{350}\right)^{-1}$
  4 rows of the pattern 111 $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq \left(\sqrt[4]{2400}\right)^{-1}$
- *Row pattern 011* $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq \frac{6}{4!} = \frac{1}{4}$
- *Row pattern 002* $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq \frac{6}{4!} = \frac{1}{4}$
- *Row pattern 001* $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq \frac{12}{4!} = \frac{1}{2}$
- *Row pattern 000* $\rightarrow Pr_{i(j)}(\mathcal{X}_j) \coloneqq \frac{24}{4!} = 1$

Each value associated with an $|A||B||C|$ pattern refers to the set of different Tessera code fragments that can be represented by the pattern. The calculation of the numbers assigned to the row pattern is outlined in the subsequent paragraphs. To summarize the calculation, we first construct all non-isomorphic undirected graphs that match the row pattern. Later, we will provide proof that this is always 1 unless the pattern is 111. We continue by counting the number of different directed graphs that are created by a simple transformation from undirected edges to directed edges. Before we dive deeper into the explanation, we introduce the injective transformation to map a directed graph onto an undirected one.

Definition 31. *Definition of an injective transformation $\varphi(\cdot)$. Be M the domain of directed acyclic graphs and N the domain of undirected unlabeled graphs. Let φ be an injective transformation $\varphi : M \to N$ so that a directed graph $\mathcal{G} = (E, V) \in M$ is mapped onto a $\varphi(\mathcal{G}) \to \dot{\mathcal{G}} = (\dot{E}, \dot{V}) \in N$. The transformation $\varphi(\cdot)$ maps each $a \to b \in E$ to an undirected edge $a - b \in \dot{E}$. Additionally, the transformation anonymizes all $v \in V$ by removing the label.*

To construct all undirected unlabeled graphs which can be transformed into directed graphs that match the row pattern, we simply prove that there is only one undirected unlabeled graph for each row pattern (unless the row pattern is $|A||B||C| = 111$). We can then use the transformation $\varphi(\cdot)$ in Definition 31 and transform one arbitrary directed graph (which can easily be constructed) into the undirected one.

Lemma 3.4.1. *Let $\mathcal{X}$ be a Tessera code and $\mathcal{X}_j = \mathcal{X} \cap \mathcal{TE}_j$ with $j \in \{1, 2, 3, 4\}$. We assume that $\mathcal{X}_j = A_j \cup B_j \cup C_j$ so that the line pattern $|A_j||B_j||C_j| \neq 111$. We suggest that for all possible $\mathcal{C}_2(\mathcal{X}_j)$ the results of the transformations $\varphi(\mathcal{C}_2(\mathcal{X}_j))$ are isomorphic.*

Proof. Let us consider the components of the $\mathcal{C}_2(\mathcal{X}_j)$ separately: $\mathcal{C}_2(A_j)$, $\mathcal{C}_2(B_j)$ and $\mathcal{C}_2(C_j)$. Let $D \in \{A_j, B_j, C_j\}$. If $|D| \in \{0, 2\}$, $\varphi(\mathcal{C}_2(D))$ always has the same result for any D. In these cases, $\varphi(\mathcal{C}_2(D))$ has to be an isomorphism to the graph in Figure 3.14 if $|D| = 0$ without the green edges and if $|D| = 2$ with the green edges. This also proves that for all row patterns where $|A|$, $|B|$, $|C| \in \{0, 2\}$, all $\varphi(\mathcal{C}_2(A \cup B \cup C))$ are isomorphisms to each other.

Hence, only row patterns which have at least one value of 1 can be represented in two different $\mathcal{C}_2$ so that their transformation $\varphi(\mathcal{C}_2)$ result would not be isomorphic. Yet, as Figure 3.15 shows, if only one value of $|A|$, $|B|$ or $|C|$ is 1, all $\varphi(\mathcal{C}_2)$ must be isomorphic. The same is true when exactly two values of $|A|$, $|B|$ or $|C|$ are 1. In that case, Figure 3.16 shows that all $\varphi(\mathcal{C}_2)$ must be isomorphic. This proves that for all row patterns $|A||B||C| \neq 111$, all $\varphi(\mathcal{C}_2(A \cup B \cup C)))$ are isomorphisms to each other.

□

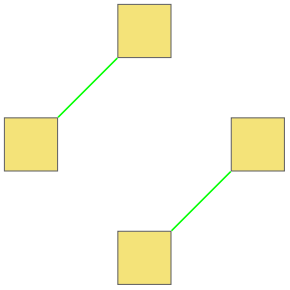


Figure 3.14: $\varphi(\mathcal{C}_2(\mathcal{X}_j))$ if $|A||B||C| = 00X$ with $X \in \{0, 2\}$. The green edges represent the value of X

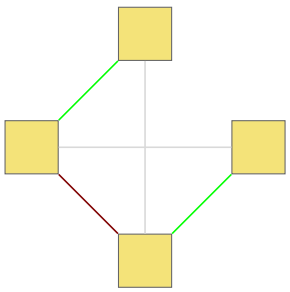


Figure 3.15: $\varphi(\mathcal{C}_2(\mathcal{X}_j))$ if $|A||B||C| = 1X_1X_2$ with $X_1, X_2 \in \{0, 2\}$. The green edges represent the value of X_1, and the grey edges represent the value of X_2

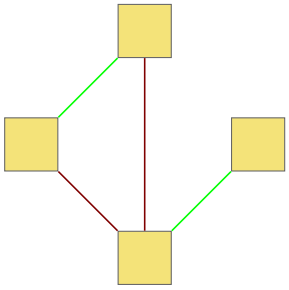


Figure 3.16: $\varphi(\mathcal{C}_2(\mathcal{X}_j))$ if $|A||B||C| = 11X$ with $X \in \{0, 2\}$. The green edges represent the value of X

Lemma 3.4.2. *Let $\mathcal{X}$ be a Tessera code and $\mathcal{X}_j = \mathcal{X} \cap \mathcal{TE}_j$ with $j \in \{1, 2, 3, 4\}$. We assume that $\mathcal{X}_j = A_j \cup B_j \cup C_j$ so that the row pattern $|A_j||B_j||C_j| = 111$. We propose that for all possible $\mathcal{C}_2(\mathcal{X}_j)$, the transformation $\varphi(\mathcal{C}_2(\mathcal{X}_j))$ has two non-isomorphic results.*

Proof. In Figure 3.17 and Figure 3.18, it can be seen that the row pattern 111 can be mapped onto non-isomorphic undirected graphs. It is obvious that the two graphs in Figure 3.17 and Figure 3.18 are the only two non-isomorphic possible results of $\varphi(\mathcal{C}_2(\mathcal{X}_j))$. □

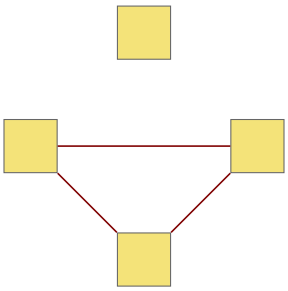


Figure 3.17: $\varphi(\mathcal{C}_2(\mathcal{X}_j))$ if $|A||B||C| = 111$

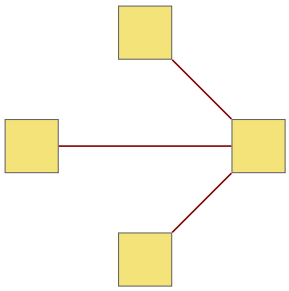


Figure 3.18: $\varphi(\mathcal{C}_2(\mathcal{X}_j))$ if $|A||B||C| = 111$

For any row pattern $|A||B||C| \neq 111$, all possible circular fragments $\mathcal{X}_i$ have one isomorphic $\varphi(\mathcal{C}_2(\mathcal{X}_i))$ for all possible $\mathcal{C}_2(\mathcal{X}_i)$. To calculate the number of possible codes which are represented by the row pattern, one can now transform $\varphi(\mathcal{C}_2(\mathcal{X}_i))$

back into directed graphs as shown in Figure 3.19. The number of possible directed acyclic graphs indicates the number of the 4! different tournaments that are actually representing different codes.

Example 31.1. *Let $|A||B||C| = 002$ be a row pattern. Figure 3.19 shows that the graph $\varphi(\mathcal{C}_2(\mathcal{X}_j))$ can be used to construct four different directed graphs. Thus, this row pattern represents four different code fragments.*

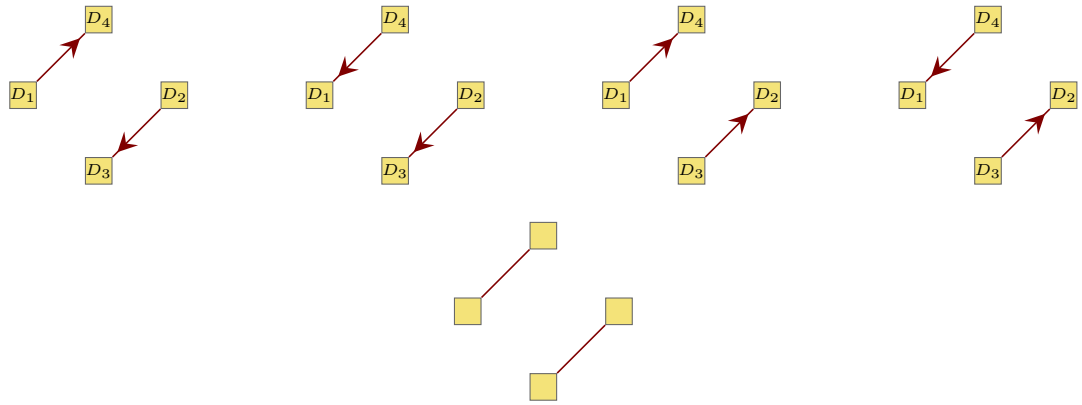


Figure 3.19: Transformation from $\varphi(\mathcal{C}_2(\mathcal{X}_j))$ into all possible $\mathcal{C}_2(\mathcal{X}_j)$ for $|A||B||C| = 002$

Using the example in Figure 3.19, one can now calculate $Pr_{i(j)}(\mathcal{X})$ for a row $j \in \{1, 2, 3, 4\}$. Let us assume that $\mathcal{T}$ is a table which represents a Tessera code. Referring to the example in Figure 3.19, we assume that the row pattern of row j is 002. $\varphi(\mathcal{C}_2(\mathcal{X}_j))$ can be transformed into 4 different $\mathcal{C}_2(\mathcal{X}_j)$ Hence, the number of different codes is four. In total, there are 4! different θ_i to construct the tournaments. Therefore, the chance that one of the 4! different $\theta_i \cdot \mathcal{T}$ leads to the correct 2-component representing $\mathcal{X}$ is calculated as follows:

$$Pr_{i(j)}(\mathcal{X}) = \frac{4}{4!} = \frac{1}{4} \textit{ see Definition 30}$$

A row in a table $\mathcal{T}$ which has a pattern 111 has to be handled differently. The 2-component satisfying the row pattern 111 can be mapped onto one of two non-isomorphic undirected unlabeled graphs. We will call one of them Y as in Figure 3.17 and the other one Z as in Figure 3.18. While Z represents eight codes, Y only represents six codes. Therefore, it can be said that if the pattern 111 appears once in a table $\mathcal{T}$, this row in $\mathcal{T}$ represents $\frac{6+8}{2} = 7$ codes. Consequently, it can be said that $Pr_{i(j)}(\mathcal{X}_j) = 7^{-1}$. In all the other cases, where $\mathcal{T}$ has more than one row with a 111 pattern, Table 3.9 explains the values.

#rows with row pattern 111	Possible combinations of Y & Z
1	Y or Z $Pr_{i(j)}(\mathcal{X}) \coloneqq \frac{6+8}{2}^{-1} = \frac{1}{7}$
2	YY, ZZ, ZY or YZ $Pr_{i(j)}(\mathcal{X}) \coloneqq \left(\sqrt{\frac{8^2+6^2+8\times6\times2}{4}}\right)^{-1} = \frac{1}{7}$
3	YYY, ZZZ, YYZ or ZZY $Pr_{i(j)}(\mathcal{X}) \coloneqq \left(\sqrt[3]{\frac{1}{4}\left(6^3+8^3+8\times6^2+6\times8^2\right)}\right)^{-1} = \frac{1}{\sqrt[3]{350}}$
4	$YYYZ$ or $ZZZY$ $Pr_{i(j)}(\mathcal{X}) \coloneqq \left(\sqrt[4]{\frac{1}{2}\left(8\times6^3+6\times8^3\right)}\right)^{-1} = \frac{1}{\sqrt[4]{2400}}$

Table 3.9: This table shows the interactions of multiple rows with a row pattern 111. The possible combinations of code fragments which are mapped onto Y(Figure 3.17) and Z (Figure 3.18) demand the number of codes represented by a table.

For a better understanding of Table 3.9, we will disclose the origin of the figures of row two. Let $\mathcal{T}$ be a table with two rows of the row pattern 111 so that $j_1, j_2 \in \{1,2,3,4\}$, $j_1 \neq j_2$ and $|A_{j_1}||B_{j_1}||C_{j_1}| = |A_{j_2}||B_{j_2}||C_{j_2}| = 111$ be the tow row pattern in the rows j_1 and j_2 in the table $\mathcal{T}$. A $\varphi(\mathcal{C}_2(\mathcal{X}_{j_2}))$ which satisfies the row pattern 111 can either be isomorphic to Y, which can be transformed into six acyclic $\mathcal{C}_2(\mathcal{X}_{j_2})$ or to Z, which can be transformed into eight acyclic $\mathcal{C}_2(\mathcal{X}_2)$. In short, Y represents six codes, and Z represents eight codes. For the rows j_1 and j_2, the possible combinations of Y & Z are YY, ZZ, ZY or YZ. Hence, two rows can either be $YY \coloneqq 6^2$ codes, $XX \coloneqq 8^2$ codes or $XY \coloneqq 6\times8$ codes. Since each table can only have one of the combinations at once, we combine the values as follows: $\frac{8^2+6^2+8\times6\times2}{4}$. The result of this equation is the result of the product of $Pr_{i(j_1)}(\mathcal{X}_{j_1}) \times Pr_{i(j_2)}(\mathcal{X}_{j_2})$. Consequently,

$$Pr_{i(j_1)}(\mathcal{X}) = Pr_{i(j_2)}(\mathcal{X}) = \left(\sqrt{\frac{8^2+6^2+8\times6\times2}{4}}\right)^{-1} = \frac{1}{7}$$

Observation 3.4.1. *Based on observations, a specialty are the tables in which all four rows are of the row pattern* 111*. In this case, the possible combinations of* Y *&* Z *are only* $YYYZ$ *or* $ZZZY$*. The combinations* $YZZZ$*,* $ZYZZ$*,* $ZZYZ$*,* $ZYYY$*,* $YZYY$*,*$YYZY$*,*$YZYZ$*,*$ZYZY$*,* $YZZY$*,* $ZYYZ$*,* $YYZZ$*,* $ZZYY$*,* $YYYY$ *and* $ZZZZ$ *always lead to to non-circular codes* $\mathcal{X}$ *associated with cyclic* $\mathcal{C}_1(\mathcal{X})$*.*

Observation 3.4.2. *Based on observations, a specialty are the tables in which three rows are of the row pattern* 111*. In this case, the possible combinations of* Y

& Z are only YYY, ZZZ, YYZ or ZZY. The combinations YZZ, ZYZ, ZYY and ZYZ always lead to non-circular codes $\mathcal{X}$ associated with cyclic $\mathcal{C}_1(\mathcal{X})$.

Example 31.2. *Let $\mathcal{X} \subset \mathcal{TE}$ be a circular Tessera code of size 3 and $\mathcal{T}(\mathcal{X})$ the representing table:*

1	1	0	0
1	1	0	0
|0	0	1	1
0	0	0	0

Table 3.10: Example of a code $\mathcal{X}$ of length 3. $\mathcal{X}_1 = \mathcal{X}_2 = \mathcal{X}_3 = 1$ and $\mathcal{X}_4 = 0$. $Pr_o(\mathcal{X}) = \frac{2}{4!}$, $Pr_{fac}(\mathcal{X}) = \frac{1}{4}$, $Pr_{i(1)}(\mathcal{X}) = Pr_{i(2)}(\mathcal{X}) = Pr_{i(3)}(\mathcal{X}) = \frac{12}{4!}$ and $Pr_{i(4)}(\mathcal{X}) = 1$. Hence, the probability that an arbitrarily chosen θ applied to $\mathcal{T}(\mathcal{X})$ constructs $\mathcal{X}$ is $Pr(\mathcal{X}) = \frac{1}{4} \times \frac{1}{12} \times (\frac{1}{2})^3 = \frac{1}{384}$

Comma-free codes The algorithm can be adapted to comma-free codes. Therefore, all but the values of $Pr_{ri}(\mathcal{X})$ are the same. Hence, we denote the inner probability of comma-free codes as:

$$Pr_{rcfi}(\mathcal{X}) = Pr_{fac}(\mathcal{X}) \times \prod_{j \in \{1,2,3,4\}} Pr_{cfi(j)}$$

so that probability $Pr_{cf}(\mathcal{X})$ of a comma-free code $\mathcal{X}$ that an arbitrarily chosen *parameter tuple* $\theta \in \Theta$ applied to a table $\mathcal{T}(\mathcal{X})$ representing $\mathcal{X}$ constructs the comma-free code $\mathcal{X}$.

$$Pr_{cf}(\mathcal{X}) = Pr_{rcfi}(\mathcal{X}) \times Pr_o(\mathcal{X})$$

Definition 32. *Let $|A||B||C|$ be a row pattern of a table $\mathcal{T}$ in row $j \in \{1,2,3,4\}$ and $\mathcal{X}_j$ the fragment represented in this row. Then, the following list defines the results of $Pr_{cfi(j)}(\mathcal{X}_j)$:*

- *Row pattern 222* $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq 0$
- *Row pattern 122* $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq \frac{1}{6}$
- *Row pattern 022* $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq \frac{1}{8}$
- *Row pattern 112* $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq \frac{1}{6}$
- *Row pattern 012* $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq \frac{1}{6}$

- *Row pattern 111 → Special case (see Table 3.9):*
  If $\mathcal{T}$ contains:
  $\leq$ 2 rows of the pattern 111 $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq 7^{-1}$
  3 rows of the pattern 111 $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq \left(\sqrt[3]{350}\right)^{-1}$
  4 rows of the pattern 111 $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq \left(\sqrt[4]{2400}\right)^{-1}$
- *Row pattern 011* $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq \frac{1}{6}$
- *Row pattern 002* $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq \frac{1}{4}$
- *Row pattern 001* $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq \frac{1}{2}$
- *Row pattern 000* $\rightarrow Pr_{cfi(j)}(\mathcal{X}_j) \coloneqq 1$

To identify the number of possible comma-free codes represented by the row pattern, one can transform $\varphi(\mathcal{C}_2(\mathcal{X}_j))$ back into directed graphs as shown above and in Figure 3.19. The number of possible directed acyclic graphs with a path length of 2 at most indicates how many of the 4! tournaments are actually different comma-free codes represented by the row pattern.

The number of Tessera codes

The number of codes represented by a table $\mathcal{T}$ can now be calculated by calculating $Pr(\mathcal{X})^{-1}$ for one represented code of $\mathcal{T}$:

$$Pr(\mathcal{X})^{-1} = |\bigcup_{\theta \in \Theta} \theta \cdot \mathcal{T}|$$

There are $(4!)^5$ *parameter tuples* $\theta \coloneqq (\theta_i, \theta_o)$ in the *parameter sets* Θ. $4!^4$ represents the combination of the different tournaments denoted as θ_i. The last factor 4! is the distribution of the four fragments to the four rows denoted as θ_o. Hence, a table can represent $(4!)^5$ at the most. If the maximal value is multiplied by the probability $Pr(\mathcal{X})$, we obtain the actual amount of *parameter tuples* which construct the same code $\mathcal{X}$ when applied to table $\theta \cdot \mathcal{T}$. Consequently, $Pr(\mathcal{X})^{-1}$ must be the number of all codes which can be represented by the same table $\mathcal{T}$.

In Table 3.11, all numbers of codes for the codes of size one to twelve (maximum size of a circular Tessera code) are listed. The algorithm presented in this chapter was used to calculate the numbers for the circular codes and the comma-free codes. However, the numbers for the 1-circular codes, C^4 codes and self-complementary circular codes were calculated by a brute force algorithm. Surprisingly, the highest number of codes for 1-circular codes, circular codes and C^4 codes is ten, whereby the highest number of codes for self-complementary circular codes and comma-free codes is nine. The investigation into this characteristic is still open, and it cannot

yet be explained. It is also important to note that the ratio of circular codes to self-complementary circular codes decreases steadily with the increasing size of the codes. Where 25% of size one circular codes are self-complementary, only 0.01651288% of size twelve circular codes are self-complementary. This suggests that an observation of a maximal self-complementary Tessera code in a genetic context would be a very strong evidence for the existence of circular Tessera codes.

Table 3.11: List of the number of circular codes and comma-free codes of different length. * The numbers in these columns were calculated by a brute force algorithm.

length	1-Circular codes*	Circular codes	C^4 codes*	Comma-free codes	self comp. circular*
1	48	48	48	48	12
2	1056	1056	1056	1056	72
3	14080	14048	14016	13952	304
4	126720	125544	124368	122376	996
5	811008	791952	773088	745584	2580
6	3784704	3606048	3433584	3214272	5408
7	12976128	11908800	10922112	9816960	9264
8	32440320	28230456	24577404	20952504	12708
9	57671680	46720800	37987120	30297824	13696
10	69206016	51111024	38129856	28015728	11232
11	50331648	33113472	22240992	14790144	6144
12	16777216	9592512	5685408	3351232	1584

The effortless and quick construction of maximal Tessera codes led to the observation that maximal circular Tessera codes have a potential link to comma-free and circular trinucleotide codes. For this observation, we used the mapping function $cod(\cdot)$ introduced in Figure 3.3 (see [40]).

Observation 3.4.3. *By applying the mapping function* $cod(\cdot)$ *to convert Tesserae to codons, we were able to observe that* 103024 *of the* 9592512 *maximal circular Tessera codes are mapped onto a circular trinucleotide code, i.e. with a probability of* 0.0107 *and* 1264 *are mapped onto a comma-free trinucleotide code, i.e. with a probability of* $\approx 10^{-4}$. 168 *of the* 1584 *maximal self-complmentary circular Tessera codes are mapped onto a circular trinucleotide code, i.e. with a probability of* 0.106. *This is almost exactly ten times higher than the probability of the* 9592512 *maximal circular Tessera codes.*

We would like to point out one special self-complementary Tessera code before

we end this chapter.

$$\mathcal{X}_T = \{AACC,\ GGTT,\ GGCC,\ AATT,\ AGTC,\ GACT,\ GATC,\ AGCT\}$$

The trinucleotide code which is the result of the mapping function $\mathcal{X} = cod(\mathcal{X}_T)$ is $\mathcal{X} \coloneqq \{GTT,\ ATT,\ GCT,\ ACT,\ GTC,\ ATC,\ GCC,\ ACC\}$. Both, the trinucleotide code $\mathcal{X}$ and Tessera code $\mathcal{X}_T$ are strong comma-free. Additionally, is the code $\mathcal{X}$ a subset of the RNY code and the complementary reversed code of the RRY code (see section 1.2.8 for RNY & RRY; [69, 11]).

3.5 Summary of Chapter 3

In sum, one can say that both circular code theory and Tessera code theory are not in conflict with each other. Rather, it seems as if the two theories favor one another. Let us first summarize some results concerning the foundation of circular Tessera codes. Firstly, there are exactly 48 of the 64 Tesserae grouped in twelve complete circular permutation classes. Hence, a circular Tessera code is called maximum if it contains exactly twelve words. Furthermore, the statements in Theorem 3.2.1 hold for any circular Tessera code $\mathcal{X} \subset \mathcal{TE}$:

(i) The maximal length of a cycle in $\mathcal{C}_1(\mathcal{X})$ is 2.

(ii) The maximal length of a path that does not contain a cycle is also 2.

(iii) The maximal length of a cycle in $\mathcal{C}_2(\mathcal{X})$ is 4. In particular, the maximal length of a path that does not contain a cycle is 3.

Recalling the definition of the reading-frame number (see Theorem 2.1.4), the shorter the length of the longest path length in a graph associated with a code, the higher the reliability regarding frameshift robustness. Note that the adaptation of the reading-frame number to Tessera codes is still open. Additionally, each circular Tessera code can be separated into four fragments by the means of $\mathcal{V}$. Let $\mathcal{X} \subset \mathcal{TE}$ be a circular Tessera code, then the following holds:

$$X = \mathcal{X}_I \cup \mathcal{X}_{SW} \cup \mathcal{X}_{YR} \cup \mathcal{X}_{KM}$$
$$\text{where } |\mathcal{X}_I| \leq 6, |\mathcal{X}_{SW}| \leq 6, |\mathcal{X}_{YR}| \leq 6, |\mathcal{X}_{KM}| \leq 6$$

By applying the mapping function $cod(\cdot)$ (see Figure 3.3) on each possible circular fragment $\mathcal{X}_\beta \in \{\mathcal{X}_I, \mathcal{X}_{SW}, \mathcal{X}_{YR}, \mathcal{X}_{KM}\}$ of each circular Tessera code, it was proven that $\mathcal{X} = cod(\mathcal{X}_\beta)$ is always a comma-free code. Furthermore, $\alpha_1(\mathcal{X})$ and $\alpha_2(\mathcal{X})$ are also comma-free codes. In section 3.3.2, the classification of graphs associated with self-complementary trinucleotide codes defined in Theorem 2.1.5 is

adapted to graphs associated with self-complementary Tessera codes. It is proven that if $\mathcal{X} \subset \mathcal{TE}$ is a circular Tessera code, $\mathcal{X}$ is self-complementary. This, however, only holds true if

1. $V = \overleftarrow{c(V)}$
2. $d^+(v) = d^-(\overleftarrow{c(v)})$ *for all vertices* $v \in V$

It should be noted that the two conditions mentioned above fit even better with the Tessera codes than with the trinucleotide codes. While the conditions only apply to trinucleotide codes of size 18 to 20, they apply to all circular Tessera codes. In addition to the properties, this chapter introduces two construction methods of circular Tessera codes. Both the method in section 3.3 and the one in section 3.4 can be used to construct all maximal circular Tessera codes. The method in section 3.4 can also be used to construct comma-free codes and circular codes of any size. Furthermore, this method was used to count all circular Tessera codes. In Table 3.11, the resulting numbers are listed. In addition, the mapping function $cod(\cdot)$ was used to shown that 103024 of the 9592512 maximal circular Tessera codes are mapped onto a circular trinucleotide code and 1264 of the 9592512 onto a comma-free trinucleotide code .

One final feature that was obtained from the construction is a tabular representation of the circular Tessera codes. All tables associated with all maximal circular Tessera codes are listed in Appendix II.2. Section 6.1 introduces a hypothesis of the evolution which includes Tessera codes. The final chapter 6 offers a hypothetical evolutionary scenario which includes Tessera codes in section 6.1.

Chapter 4

The relation between k-circularity and circularity of codes

This chapter, which refers to the published article [22], introduces a new class of block codes: the k-circular codes. k-circular codes belong to the circular code family. As a weakened version of the circular code, such a k-circular code is a frameshift error detecting code and consists exclusively of words of word length ℓ. In the sections of this chapter, we will first define k-circularity for arbitrary word lengths over arbitrary finite alphabets in section 4.1, before presenting the graph characterization of k-circularity in section 4.2. In section 4.3, the question of when k-circularity implies circularity will be posed and answered. Additionally, in section 4.3.1 the sharpness of these borders is verified . This theorem, the so called Sharpness Theorem, will then be proven in the subsequent chapter. Finally, Chapter 4 closes with the biological consequences of k-circularity in section 4.4.

4.1 Introduction and definition of k-circularity

If a circular code can find the correct reading-frame in every word written on a circle, a k-circular code requires a concatenation of a maximum of k words to reliably ensure the reading-frame. Therefore, a concatenation of $k+1$ words written on a circle could be read in more than one reading-frame. Thus, a k-circular code does not have to be $k+1$-circular but must be $k-1$-circular. The class of k-circular codes contains both circular and comma-free codes as subclasses for every given k. Hence, the definition of circular codes can be refined by their relation to k-circular codes. Let us first recall the definition of the $\mathcal{X}$-decomposition in the first bullet point in Definition 33. Subsequently, Definition 33 depicts the formal definition of the k-circular codes, thus refining the definition of the circular codes.

Definition 33. *Let $\mathcal{X} \subseteq \Sigma^\ell$ be an ℓ-letter code.*

- *For $w \in \Sigma^*$, an* $\mathcal{X}$-decomposition *of w is a word $w_1 \cdots w_r \in \mathcal{X}^r$ with $r \in \mathbb{N}$ such that $w = w_1 \cdot w_2 \cdots w_r$.*

- *Let m be a positive integer and let $b_1 \cdots b_m \in \mathcal{X}^m$. A* circular $\mathcal{X}$-decomposition *of the concatenation $w := b_1 \cdots b_m$ is an $\mathcal{X}$-decomposition of a circular permutation of $\alpha_j(w)$ for some $j \in \{1, \ldots, m-1\}$.*

- *Let k be a positive integer. The code $\mathcal{X}$ is* k-circular *if for every $m \in \{1, \ldots, k\}$ and every m-words $w_1, \ldots, w_m$ of words in $\mathcal{X}$, the concatenation $w_1 \cdots w_m$ admits a unique circular $\mathcal{X}$-decomposition.*

- *The code $\mathcal{X}$ is* circular *if it is k-circular for all $k \in \mathbb{N}$.*

- *We define $k(\mathcal{X})$ to be ∞ if $\mathcal{X}$ is circular and the least integer r such that $\mathcal{X}$ is not r-circular otherwise.*

4.2 Graph characterization of k-circularity

Directed cyclic paths in the graph associated with a code are of vast importance (see Definition 9) in the research of circular codes. A cyclic graph forces the code associated with this graph to be a non-circular code. In the case of k-circular codes, the restrictions are not as stringent (see Definition 33). Hence, while studying the k-circular codes, these cyclic paths began to move into focus. As a result of the examination of cyclic paths, we were able to make the following observations.

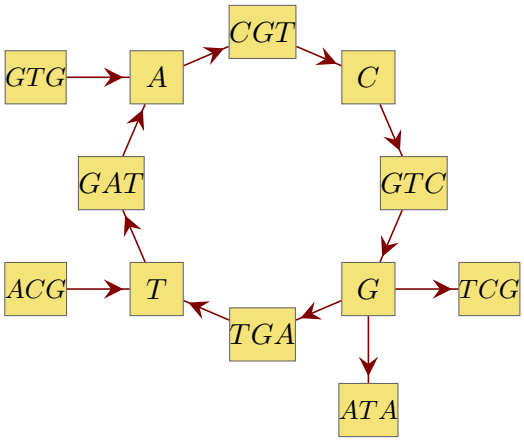


Figure 4.1: The 2-component $\mathcal{C}_2(\mathcal{X})$ of the tetranucleotide code $\mathcal{X} = \{CGTC, GTCG, GTGA, TGAT, GATA, ACGT\}$. The cyclic path cannot be extended, since all available 1-nodes are already included.

Observation 4.2.1. *Given two integers ℓ and n, both at least 2, let $\mathcal{X}$ be an ℓ-letter code over an alphabet Σ of cardinality n. Most observations can be traced in Figure 4.1.*

1. *For every $j \in \{1, \ldots, \lfloor \ell/2 \rfloor\}$, the j-component of $\mathcal{G}(\mathcal{X})$ contains no more than $n^j + n^{\ell-j}$ vertices if $j \neq \ell/2$ and no more than $n^{\ell/2}$ otherwise.*

2. *If ℓ is odd, $\mathcal{G}(\mathcal{X})$ contains no directed cycle of odd length.*

3. *If ℓ is even, every directed cycle of odd length in $\mathcal{G}(\mathcal{X})$ is contained in the $(\ell/2)$-component.*

4. *Suppose that $\mathcal{G}(\mathcal{X})$ contains a directed cycle of length t and let $j \in \{1, \ldots, \lfloor \ell/2 \rfloor\}$ such that this cycle is contained in the j-component. Then,*

$$t \leq \begin{cases} 2n^j & \text{if } j \neq \frac{\ell}{2}, \\ n^{\ell/2} & \text{if } j = \frac{\ell}{2}. \end{cases}$$

In article [27], the k-circularity of dinucleotide codes was first characterized in terms of graphs:

- a dinucleotide code is 1-circular but not 2-circular if and only if its associated graph contains a Hamiltonian cycle of length 4;
- a dinucleotide code is 2-circular but not 3-circular if and only if its associated graph contains an oriented cycle of length 3 and has no Hamiltonian cycle; and
- a dinucleotide 3-circular code is circular.

These observations led to Theorem 4.2.1. This theorem is a natural generalization of the approach from article [27]. For this generalization the following notations must be defined:

Definition 34. *If $\mathcal{X}$ is a code, then:*

$g_o(\mathcal{X})$ *denotes the respective lengths of the shortest directed cycles of odd length. We define $g_o(\mathcal{X}) \coloneqq \infty$ if no cycles of odd length exist in $\mathcal{G}(\mathcal{X})$,*

$g_e(\mathcal{X})$ *denotes the respective lengths of the shortest directed cycles of even length. We define $g_e(\mathcal{X}) \coloneqq \infty$ if no cycles of even length exist in $\mathcal{G}(\mathcal{X})$,*

Thus, if the code $\mathcal{X}$ is non-circular, one of these numbers must be finite.

Theorem 4.2.1. *(Theorem 3.3, [22]) Let Σ be a finite alphabet, ℓ an integer greater than 1 and $\mathcal{X} \subseteq \Sigma^\ell$ an ℓ-letter code over Σ. Then the code $\mathcal{X}$ is k-circular and not $(k+1)$-circular if and only if*

$$k = \min\left\{g_o(\mathcal{X}), \frac{g_e(\mathcal{X})}{2}\right\} - 1 < \infty.$$

Proof. First, suppose that $w_1 \cdots w_r$ is a directed cycle in $\mathcal{G}(\mathcal{X})$. If r is even, the word $w_1 \cdots w_r$ admits two different circular $\mathcal{X}$-decompositions into $r/2$ words from $\mathcal{X}$, namely

$$w_1w_2|\ldots|w_{r-1}w_r \quad \text{and} \quad w_2w_3|\ldots|w_{r-2}w_{r-1}|w_rw_1.$$

It follows that $\mathcal{X}$ is not $\frac{1}{2}g_e(\mathcal{X})$-circular. If r is odd, then the word $w_1 \cdots w_r w_1 \cdots w_r$ admits two different circular $\mathcal{X}$-decompositions into r words from $\mathcal{X}$, namely

$$w_1w_2|\ldots|w_{r-2}w_{r-1}|w_rw_1|w_2w_3|\ldots|w_{r-1}w_r \quad \text{and}$$
$$w_2w_3|\ldots|w_{r-1}w_r|w_1w_2|\ldots|w_{r-2}w_{r-1}|w_rw_1.$$

It follows that $\mathcal{X}$ is not $g_o(\mathcal{X})$-circular.

Conversely, suppose that both w and its circular j-permutation admit a circular $\mathcal{X}$-decomposition where $j \in \{1, \ldots, \ell-1\}$. By setting $a_{r+1} \coloneqq a_1$, the word w can be written $a_1b_1|\ldots|a_rb_r$ such that $|a_i| = j$, $|b_i| = \ell - j$, and $a_ib_i, b_ia_{i+1} \in \mathcal{X}$ for each $i \in \{1, \ldots, r\}$. It follows from Definition 9 that

$$W \coloneqq a_1 \to b_1 \to \ldots \to a_r \to b_r \to a_1$$

is a closed walk in $\mathcal{G}(\mathcal{X})$. Consequently, W either contains a directed cycle of even length, which then must be of length $|V(W)| = 2r$ at most or W decomposes into an even number of directed odd cycles, one of them thus having length r at the most. Consequently, if $\mathcal{X}$ is not r-circular, then $g_e(\mathcal{X}) \leq 2r$ or $g_o(\mathcal{X}) \leq r$. □

We would like to remark that the results from [27] are consistent with the statement of Theorem 4.2.1 above.

- If $\mathcal{X}$ is a 2-letter code that is 1-circular but not 2-circular, then $\mathcal{G}(\mathcal{X})$ contains cycles of lengths 3 and 4, and, indeed, $1 = \min\left\{3, \frac{4}{2}\right\} - 1$.

- If $\mathcal{X}$ is a 2-letter code that is 2-circular but not 3-circular, then $\mathcal{G}(\mathcal{X})$ contains a cycle of length 3 and no cycle of even length, and, indeed, $2 = \min\{3, \infty\} - 1$.

In conclusion, in Observation 4.2.1, a cycle in $\mathcal{G}(\mathcal{X})$ is bound by the cardinality n of the alphabet and the word length ℓ. Additionally, Theorem 4.2.1 points out that a k-circular but non-circular code is represented by a cyclic graph. In combination, these two facts imply that there must be a k-circularity depending on n and ℓ so that $k+1$-circularity is no longer possible. Simplified, there must be a value $k = k(n, \ell)$ so that $k+1$-circularity implies circularity. In the following section, we will introduce our most important theorems, which completely determine this barrier.

4.3 When does k-circularity imply circularity?

To determine whether a code is circular, one needs to verify if any concatenation of infinite words of a code written on a cycle can only be decomposed in words of the code in one reading-frame. Consequently, a code is circular if and only if it is a ∞-circular code (see Definition 33). This causes it to be an undecidable problem. To convert this problem into a decidable one, it is necessary to identify a number $0 < k(n, \ell) < \infty$ so that a code is only circular if it is $k(n, \ell)$-circular where n is the cardinal number of the alphabet and ℓ is the word length of the code. Hence, the issue in focus of this section is the identification of an integer $k(n, \ell)$ so that $k(n, \ell)$-circular is equally circular. We first would like to mention that the cases where n or ℓ is 1 are trivial, with Theorem 4.3.1 providing a full answer to this question. Subsequently, Theorem 4.3.2 demonstrates that the boundary of $k(n, \ell)$ is sharp.

Theorem 4.3.1. *(Theorem 4.1, [22]) Let both ℓ and n be integers of at least 2. Let Σ be an alphabet with $|\Sigma| = n$ and $\mathcal{X} \subseteq \Sigma^\ell$ an ℓ-letter code over Σ. Set*

$$k(n, \ell) \coloneqq \begin{cases} n^{\frac{\ell-1}{2}} & \text{if } \ell \text{ is odd,} \\ n^{\ell/2} & \text{if } \ell \text{ is even and } n \text{ is odd,} \\ n^{\ell/2} - 1 & \text{if } \ell \text{ is even and } n \text{ is even.} \end{cases}$$

Then, the code $\mathcal{X}$ is circular if and only if it is $k(n, \ell)$-circular.

Proof. One direction is trivial: if $\mathcal{X}$ is circular, then $\mathcal{X}$ is r-circular for every integer r, and hence, for $k(n, \ell)$.

Conversely, let $\mathcal{X}$ be an ℓ-letter code that is r-circular but not $(r+1)$-circular. According to Theorem 4.2.1, the graph $\mathcal{G}(\mathcal{X})$ contains a cycle of (even) length $2(r+1)$, or r is even and $\mathcal{G}(\mathcal{X})$ contains a cycle of (odd) length $r+1$.

(i) If ℓ is odd, then Observation 4.2.1 yields that $g_o(\mathcal{X}) = \infty$. Therefore, in this case, $r + 1 = \frac{1}{2} g_e(\mathcal{X}) < g_o(\mathcal{X})$. Because ℓ is odd, Observation 4.2.1 implies that

$$g_e(\mathcal{X}) \leq \max\left\{2n^j \,:\, 1 \leq j \leq (\ell-1)/2\right\} = 2n^{\frac{\ell-1}{2}}.$$

Consequently, $r+1 \leq n^{(\ell-1)/2}$ and hence, $r \leq k(n,\ell)-1$.

(ii) Assume now that ℓ is even. Note that if $r+1 = \frac{1}{2}g_e(\mathcal{X})$, then according to Observation 4.2.1,

$$r+1 \leq \max\left\{\frac{1}{2}\cdot n^{\ell/2}, n^{\ell/2-1}\right\} \leq n^{\ell/2}-1.$$

Further suppose that $r+1 = g_o(\mathcal{X}) < \frac{1}{2}g_e(\mathcal{X})$. Observation 4.2.1 implies that $g_o(\mathcal{X}) \leq n^{\ell/2}$, with equality only if n is odd, since $g_o(\mathcal{X})$ is odd. Therefore,

$$r \leq \begin{cases} n^{\ell/2}-1 & \text{if } n \text{ is odd (and } \ell \text{ is even)},\\ n^{\ell/2}-2 & \text{if } n \text{ is even (and } \ell \text{ is even)}. \end{cases}$$

We established that whenever

$$r \geq \begin{cases} n^{\frac{\ell-1}{2}} & \text{if } \ell \text{ is odd},\\ n^{\ell/2} & \text{if } \ell \text{ is even and } n \text{ is odd},\\ n^{\ell/2}-1 & \text{if } \ell \text{ is even and } n \text{ is even}, \end{cases}$$

every ℓ-letter code over an alphabet of cardinality n that is r-circular must be circular. This concludes the proof. □

We further apply Theorem 4.3.1 to the case of a binary alphabet, *i.e.* with $n = 2$, and obtain the following statement:

Observation 4.3.1. *(Corollary 4.3, [22]) Let $\Sigma = \{0,1\}$ and ℓ be an integer greater than 1. A binary ℓ-letter code $\mathcal{X} \subseteq \Sigma^\ell$ is circular if and only if $\mathcal{X}$ is*

1. *$2^{\frac{\ell-1}{2}}$-circular if ℓ is odd; or*
2. *$(2^{\frac{\ell}{2}}-1)$-circular if ℓ is even.*

Including the proof of Theorem 4.3.1, let us note that it implies the following results with dinucleotide and trinucleotide circular codes over the genetic alphabet, which were previously cited [29, 30]:

Observation 4.3.2. *(Corollary 4.2, [22])*

1. *A trinucleotide code $\mathcal{X} \subseteq \mathfrak{B}^3$ is circular if and only if $\mathcal{X}$ is 4-circular.*
2. *A dinucleotide code $\mathcal{X} \subseteq \mathfrak{B}^2$ is circular if and only if $\mathcal{X}$ is 3-circular.*

Proof.

(1) In this case, $n = 4$ and $\ell = 3$. Hence, according to Theorem 4.3.1, the code $\mathcal{X}$ is circular if and only if it is $4^{\frac{3-1}{2}} = 4^1 = 4$-circular.

(2) In this case, $n = 4$ and $\ell = 2$. Hence, according to Theorem 4.3.1, the code $\mathcal{X}$ is circular if and only if it is $(4^{\frac{2}{2}} - 1) = 3$-circular.

□

Subsequently, we present codes that are $(k(n,\ell) - 1)$-circular but non-circular for some specific values of n and ℓ. In the subsequent section 4.3.1 the remaining question of whether or not the bounds given in the Theorem 4.3.1 are sharp is answered.

Example 34.1. *The following examples originate from the constructions presented in Section 4.3.1. These examples are minimal $(k(n,\ell) - 1)$-circular. Additionally, they are minimal regarding to Theorem 4.2.1 as we will explain at the beginning of section 4.3.1.*

1. *Let $n = 2$ and $\ell = 3$. Every 2-circular 3-letter code is circular and there are 1- but not 2-circular 3-letter codes:*

$$\mathcal{X}_{(2,3)} = \{010,\ 101\}$$

2. *Let $n = 2$ and $\ell = 4$. Every 3-circular 4-letter code is circular and there are 2- but not 3-circular 4-letter codes over $\{0,1\}$:*

$$\mathcal{X}_{(2,4)} = \{0001,\ 0111,\ 1100\}$$

3. *Let $n = 2$ and $\ell = 5$. Every 4-circular 5-letter code is circular and there are 3- but not 4-circular 5-letter codes:*

$$\mathcal{X}_{(2,5)} = \{00100,\ 01110,\ 10001,\ 11011\}$$

4. *Let $n = 2$ and $\ell = 7$. Every 8-circular 7-letter code is circular and there are 7- but not 8-circular 8-letter codes:*

$$\mathcal{X}_{(2,7)} = \{0001000,\ 0010010,\ 0101100,\ 0111110,\ 1000001,\ 1011011,\ 1100101,\ 1110111\}$$

5. *Let $n = 3$ and $\ell = 4$. Every 9-circular 4-letter code is circular and there are 8- but not 9-circular 4-letter codes over $\{0,1,2\}$:*

$$\mathcal{X}_{(3,4)} = \{0001,\ 0102,\ 0210,\ 1011,\ 1120,\ 2021,\ 2122,\ 2200\}$$

6. *Let $n = 4$ and $\ell = 3$ (that is, trinucleotides over the genetic alphabet). Every 4-circular 3-letter code is circular and there are 3- but not 4-circular 3-letter code over $\mathfrak{B}$:*

$$\mathcal{X}_{(4,3)} = \{AGC,\ ATT,\ CAA,\ CTG,\ GCC,\ GAT,\ TCA,\ TGG\}$$

7. *Let $n = 4$ and $\ell = 4$ (that is, tetranucleotides over the genetic alphabet). Every 15-circular 4-letter code is circular and there are 14- but not 15-circular 4-letter codes over $\mathfrak{B}$:*

$$\mathcal{X}_{(4,4)} = \{AAAC,\ ACAG,\ AGAT,\ ATCA,\ CACC,\ CCCG,\ CGCT,\ CTGA,\ GAGC,\ GCGG,\ GGGT,\ GTTA,\ TATC,\ TCTG,\ TGAA\}$$

4.3.1 Proof of the sharpness of Theorem 4.3.1

This section provides evidence that the boundaries established in theorem 4.3.1 are sharp. We prove this by constructing a code for an arbitrary alphabet Σ and a word length $\ell > 2 \in \mathbb{N}$ which is $k(n,\ell) - 1$-circular but not circular where $n = |\Sigma|$. Thus, it can be said that $k(n,\ell) - 1$ is the smallest integer so that a code $\mathcal{X} \subseteq \Sigma^\ell$ which is $k(n,\ell) - 1$-circular does not need to be circular. The obtained results in Theorem 4.2.1 illustrate that a graph associated with a $k(n,\ell) - 1$-circular code must contain a cycle that has a length of $k(n,\ell)$ if $k(n,\ell)$ is odd and a length of $2 \times k(n,\ell)$ otherwise. Therefore, we construct a code with a unique cycle of the required length so that it guarantees that such a code is $(k(n,\ell)-1)$-circular but not circular. We divide the demonstration into three cases: if ℓ is even (Lemma 4.3.1) and if ℓ is odd (Lemma 4.3.2). The case that ℓ is odd is again divided into two cases: if ℓ is odd and $n = 2$ (Lemma I.1) and finally, if ℓ is odd and $n \geq 3$ (Lemma I.2). The proofs of Lemmas I.1 and I.2 are quite long and complex and therefore not included in this Thesis. They can be found in article [22](Lemma I.1 and I.2). However, there are new mathematical tools for the analysis of codes that may be able to identify additional properties in the genetic code in the future. Let us give the main idea of the constructions.

Theorem 4.3.2. *(Theorem 4.4, [22]) Given two integers n and ℓ both at least 2. Let Σ be an alphabet with $|\Sigma| = n$ and $\mathcal{X} \subseteq \Sigma^\ell$ an ℓ-letter code over Σ. Then, $k(n,\ell)$ is the least integer r such that every code $\mathcal{X} \subseteq \Sigma^\ell$ that is r-circular is circular.*

We begin with the case in which ℓ is even.

Lemma 4.3.1. *(Lemma 5.1, [22]) If n and ℓ are integers both at least 2 and ℓ is even, then there is an ℓ-letter code over Σ that is $(k(n,\ell) - 1)$-circular. Yet, this ℓ-letter code over Σ is non-circular when $k(n,\ell) := n^{\ell/2}$ if n is odd and $k(n,\ell) := n^{\ell/2} - 1$ if n is even.*

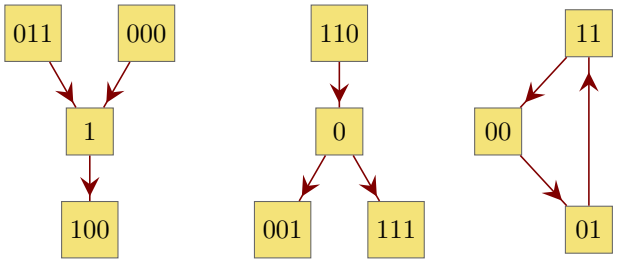


Figure 4.2: The graph $\mathcal{G}(\mathcal{X}_{(2,4)})$ associated with the binary 4-letter code $\mathcal{X}_{(2,4)} = \{0001, 1100, 0111\}$.

Proof. Note that $k(n,\ell)$ is always odd in this setting. The situation in which $n = 2 = \ell$ is trivial: every code is 0-circular, and the code $\{00\}$ is not 1-circular.

We now assume that $(n,\ell) \neq (2,2)$. We construct an ℓ-letter code $\mathcal{X}(n,\ell)$ over Σ_n such that $\mathcal{G}(\mathcal{X}(n,\ell))$ contains a unique cycle, which is of odd length $k(n,\ell)$. It thus follows that $\mathcal{X}(n,\ell)$ is $(k(n,\ell)-1)$-circular but not $k(n,\ell)$-circular by Theorem 4.2.1. The code is constructed by using n-adic representations of numbers below $n^{\ell/2}$ (see Equation (4.1)).

First, we rule out a boundary case: if $n = 2$ and $\ell = 4$, then one directly verifies that $\mathcal{X}_{(2,4)} := \{0001, 1100, 0111\}$ is 2-circular but not 3-circular. Indeed, the associated graph $\mathcal{G}(\mathcal{X}_{(2,4)})$, depicted in Figure 4.2, contains a unique directed cycle, which has length 3.

From now on, we assume that $(n,\ell) \neq (2,4)$ so that $n \geq 3$ or $\ell \geq 6$. Given a length $i \leq \ell/2$ and an integer $x < n^i$, we define $(x)_i$ to be the word of length i over the alphabet Σ_n representing x written in basis n. For example, if $n = 3$, then $(8)_4 = 0022$. If $w = (x)_i$, we also write $x = [\![w]\!]$. To improve readability, we use Y and Z to refer to $n-2$ and $n-1$, respectively, when writing words in Σ_n^*.

Let $\mathcal{X}(n,\ell)$ be defined as

$$\mathcal{X}(n,\ell) := \left\{(x)_{\ell/2} \cdot (x+1)_{\ell/2} \ : \ x \in \mathbf{Z}_{k(n,\ell)}\right\} \tag{4.1}$$

where the addition is in $\mathbf{Z}_{k(n,\ell)}$. Note that $|\mathcal{X}| = k(n,\ell)$. For instance, if $n = 2$ and $\ell = 6$, then $k(n,\ell) = n^{\ell/2} - 1 = 7$, and

$$\mathcal{X}_{(2,6)} = \{000001, 001010, 010011, 011100, 100101, 101110, 110000\}.$$

We assert that the graph $\mathcal{G}(\mathcal{X}(n,\ell))$ associated with $\mathcal{X}(n,\ell)$ has a unique cycle of odd length $k(n,\ell)$. It then follows from Theorem 4.2.1 that the ℓ-letter code $\mathcal{X}(n,\ell)$ is $(k(n,\ell)-1)$-circular but not $k(n,\ell)$-circular.

We begin by proving a useful assertion on the code $\mathcal{X}^-(n,\ell)$ defined as

$$\mathcal{X}^-(n,\ell) := \left\{(x)_{\ell/2} \cdot (x+1)_{\ell/2} \ : \ x \in \mathbf{Z}_{k(n,\ell)} \setminus \{k(n,\ell)-1\}\right\}.$$

A. *The graph* $G^- \coloneqq \mathcal{G}(\mathcal{X}^-(n,\ell))$ *is acyclic. Moreover, for every* $i \in \{1,\dots,\ell/2-1\}$*, every directed path of length* 2 *in* G^- *with a middle* $(\ell-i)$*-node begins at the* i*-node* Z^i *and ends at the* i*-node* 0^i*. In particular, all out-neighbours of* 0^i *in* G^- *have an out-degree of* 0*, and all in-neighbours of* Z^i *in* G^- *have an in-degree of* 0*.*

Proof. For each $i \in \{1,\dots,\ell/2\}$, let C_i be the i-component of G^-. Note that $C_{\ell/2}$ is a path of length $k(n,\ell)-1$ for $x \in \mathbf{Z}_{k(n,\ell)}$, traversing all nodes $(x)_{\ell/2}$ in increasing order. It follows that $C_{\ell/2}$ is acyclic. We now fix $i \in \{1,\dots,\ell/2-1\}$ and focus our attention on the component C_i, which has no odd cycle. Let $y = y_1 \cdot y_2 \cdot y_3$ be an $(\ell-i)$-node of G where $|y_1| = |y_3| = \ell/2 - i$ and $|y_2| = i$. Let us prove that if G^- contains two nodes x and x' such that $x \to y \to x'$, then $x = Z^i$ and $x' = 0^i$.

1. By the definition of G^-, if there is an arc from a node x to y, then x is an i-node, and it holds in $\mathbf{Z}_{k(n,\ell)}$ that

   $$[\![y_2 \cdot y_3]\!] = [\![x \cdot y_1]\!] + 1.$$

   There are two possible cases:

   (a) $y_2 = x$ and $[\![y_3]\!] = [\![y_1]\!] + 1$; or
   (b) $[\![y_2]\!] = [\![x]\!] + 1$, $y_1 = Z^{\ell/2-i}$ and $y_3 = 0^{\ell/2-i}$.

   Note that $y_1 \neq y_3$ in both cases.

2. On the other hand, if there is an arc from y to a node x', then x' is an i-node, and it holds in $\mathbf{Z}_{k(n,\ell)}$ that

   $$[\![y_3 \cdot x']\!] = [\![y_1 \cdot y_2]\!] + 1.$$

   Again, there are two possible cases:

   (a) $y_3 = y_1$ and $[\![x']\!] = [\![y_2]\!] + 1$; or
   (b) $[\![y_3]\!] = [\![y_1]\!] + 1$, $y_2 = Z^i$ and $x' = 0^i$.

Let us assume that y has both an ingoing arc $x \to y$ and an outgoing arc $y \to x'$ where it might be that $x = x'$. Because of the arc $x \to y$, we know that $y_1 \neq y_3$. Thus, case 2a is impossible, and we consequently must fall into case 2b. Therefore, $x' = 0^i$, $[\![y_3]\!] = [\![y_1]\!] + 1$, and we infer that the arc $x \to y$ fell into case 1a, so $x = y_2 = Z^i$. This proves the "moreover" part of the assertion. It directly follows that 0^i does not belong to a directed cycle in C_i, and consequently no $(\ell-i)$-node belongs to a directed cycle in C_i. Because all arcs in C_i are between an i-node and an $(\ell-i)$-node, we deduce that C_i is acyclic, thereby concluding the proof of (A).

We now show that $\mathcal{G}(\mathcal{X})$ contains exactly one cycle which spans its whole component $C_{\ell/2}$. The code $\mathcal{X}$ is obtained from $\mathcal{X}^-$ by adding the word $Z^{\ell/2-1}Y0^{\ell/2}$ if n is even or the word $Z^{\ell/2}0^{\ell/2}$ if n is odd. Let us see how $\mathcal{G}(\mathcal{X})$ is obtained from G^-. As mentioned earlier, the $\ell/2$-component of G^- is a path of length $n^{\ell/2}-1$ traversing all nodes $(x)_{\ell/2}$ for $x \in \mathbf{Z}_{k(n,\ell)}$ in increasing order. Because the $\ell/2$-component of $\mathcal{G}(\mathcal{X})$ is obtained from that of G^- by adding an arc from $(k(n,\ell)-1)_{\ell/2}$ to $(0)_{\ell/2}$, we precisely obtain a cycle of length $k(n,\ell)$. For each $i \in \{1,\ldots,\ell/2-1\}$, the component C_i is obtained from the i-component of G^- by adding an arc outgoing from the i-node Z^i and an arc ingoing to the i-node 0^i. This does not create a cycle of a length of at least 4, since G^- contains no directed path from 0^i to Z^i. The only cycles that might be created are therefore of length 2, and there are two possible ones. The first possibility is to create a cycle containing precisely the i-node 0^i and either the $(\ell-i)$-node $Z^{\ell/2-1}Y0^{\ell/2-i}$ if n is even or the $(\ell-i)$-node $Z^{\ell/2}0^{\ell/2-i}$ if n is odd. If this cycle is created, then

$$\begin{cases} [\![Z^{i-1}Y0^{\ell/2-i}]\!] = [\![0^iZ^{\ell/2-i}]\!] + 1 & \text{if } n \text{ is even,} \\ [\![Z^i0^{\ell/2-i}]\!] = [\![0^iZ^{\ell/2-i}]\!] + 1 & \text{if } n \text{ is odd.} \end{cases}$$

This is possible only if $i = 1$ and either $Y = 1$ and n is even or $Z = 1$ and n is odd. However, the parity of n is the same as that of $Y = n-2$ and different from that of $Z = n-1$. Therefore, this first possible cycle is not created.

The other possible cycle is the one containing the i-node Z^i and either the $(\ell-i)$-node $Z^{\ell/2-i-1}Y0^{\ell/2}$ if n is even or the $(\ell-i)$-node $Z^{\ell/2-i}0^{\ell/2}$ if n is odd. If this cycle is created, then

$$\begin{cases} [\![0^{\ell/2-i}Z^i]\!] = [\![Z^{\ell/2-i-1}Y0^i]\!] + 1 & \text{if } n \text{ is even,} \\ [\![0^{\ell/2-i}Z^i]\!] = [\![Z^{\ell/2-i}0^i]\!] + 1 & \text{if } n \text{ is odd.} \end{cases}$$

If n is odd, then the equality implies that $i = \ell/2$, which is not the case. If n is even, then the equality implies that $\ell/2 - i = 1$ and $Y = 0$, *i.e.* $n = 2$. It would follow that $[\![01^i]\!] = [\![0^{i+1}]\!] + 1$, implying $i = 1$ and hence $\ell = 4$. However, this is not the case as we assumed that $(n,\ell) \neq (2,4)$.

This completes the proof of Lemma 4.3.1. □

We now proceed with the case in which ℓ is odd. The entire proof is divided into two cases depicted in detail in Appendix I. of the article [22] (Lemma I.1 and I.2). Here we only present a sketch of the proof to illustrate its strategy and main steps.

Lemma 4.3.2. *(Lemma 5.2, [22]) Let n and ℓ be integers both at least 2 and assume that ℓ is odd. There is an ℓ-letter code over Σ_n that is $(n^{(\ell-1)/2}-1)$-circular but non-circular.*

Sketch of proof. Let us first deal with the case in which ℓ is odd and $n = 2$. We have to show that there is an ℓ-letter code over Σ_n that is $(2^{(\ell-1)/2} - 1)$-circular but non-circular. Let us fix $k \coloneqq k(2, \ell) = 2^{\frac{\ell-1}{2}}$. The aim is to construct a binary code $\mathcal{X}$ so that its graph $\mathcal{G}(\mathcal{X})$ contains a unique cycle of length $2k$. Let $s \coloneqq \frac{\ell-3}{2}$, $S \subseteq \{0,1\}^s$ be a subset of binary words of length s and set

$$\mathcal{X}_S \coloneqq \{a \cdot y \cdot a \cdot y \cdot a \ : \ a \in \{0,1\}, y \in S\} \cup \{a \cdot y \cdot \overline{a} \cdot y \cdot a \ : \ a \in \{0,1\}, y \in \{0,1\}^s \setminus S\}$$

where for a binary word w, the *complement* of w is the word $\overline{w}$ obtained from w by complementing each of its letters. It will be shown in Lemma I.1 (Appendix I. in article [22]) that all but one of the components of $\mathcal{G}(\mathcal{X})$ are acyclic if $S \subseteq 01\{0,1\}^{s-2}$, and that there is a choice for such a set S so that the remaining component consists of exactly one cycle. This solves the binary case.
The case in which ℓ is odd and $n > 2$ is more delicate. As before, one sets $s \coloneqq \frac{\ell-3}{2}$ and $k \coloneqq n^{\frac{\ell-1}{2}}$. In Lemma I.2 (Appendix I. in article [22]), we shall define a mapping φ from Σ_n^s to the family of 3-letter codes over Σ_n with very specific properties so that the associated code

$$X_\varphi \coloneqq \{a \cdot y \cdot b \cdot y \cdot c \ : \ y \in \Sigma_n^s \text{ and } abc \in \varphi(y)\}$$

satisfies our requirements. All details for the construction of φ can be found in Lemma I.2 (Appendix I. in article [22]), but we would like to notice that in the binary case, we have $\varphi(y) = \{010, 101\}$ if $y \notin S$ and $\varphi(y) = \{000, 111\}$ otherwise. □

With 4.3.1 and 4.3.2 we conclude the proof of our Theorem 4.3.2.

4.4 Biological consequences

Evidence such as the X-code indicates that circular codes and all related block codes can play a hypothetical role in the evolutionary development of the standard genetic code (SGC). However, neither the three comma-free codes *RRY, RNY* and *GNC* [45, 69, 11] nor the X-code encodes all 20 amino acids. In fact, there are $12,964,440$ maximum circular codes under the genetic alphabet and none of them encodes 20 or 19 amino acids with SGC. Only ten maximum circular codes encode 18 amino acids with SGC (see [57]). Furthermore, it is noticeable that among the $3^{20} = 3,486,784,401$ maximal 1-circular codes only 52 encode for all 20 amino acids, i.e. with a very small probability of approximately 10^{-8}, (see the list given in Appendix II.3 or [52]*Table 2 where they were called bijective genetic codes without permuted trinucleotides *WPTBGC* before the k-circular code theory was developed in this thesis). This supports the assumption that due to the expansion

of the code in evolution, the circular codes were replaced by k-circular codes. It has already been verified [29] that these 52 maximum 1-circular codes cannot be 2-circular. In addition, the following construction based on graph theory underlines once more the outstanding role of this class of 52 maximum 1-circular codes. This offers a theoretical framework for the previous calculation results [52].

In the following lemma, we will proceed with the proof of the existence of the 52 maximal 1-circular codes that encode all 20 amino acids. Before we introduce the lemma, however, we need to provide the basics of *bipartite graphs*. A *bipartite graph* is a graph $\mathcal{G} = (V, E)$ such that the set of vertices V is the disjoint union of two sets A and B such that any edge in $e \in E$ is of the form $e = (a, b)$ for some $a \in A$ and $b \in B$. Hence, no edge connects two vertices from A or two vertices from B. A *perfect matching* of $\mathcal{G}$ is a subset $M \subseteq E$, such that the edges of M form a bijection between the sets A and B. A perfect matching can thus only exist if A and B have the same cardinality.

Lemma 4.4.1. *(Lemma 6.1, [22]) There are exactly* 52 *maximal* 1*-circular codes that encode all* 20 *amino acids.*

Proof. We first need to recall some essential facts [29]. An equivalence class $[c]$ of some codon $c \in \mathfrak{B}^3$ consists of c and its circular permutations, *e.g.* $[ATC] = \{ATC, TCA, CAT\}$. The equivalence class is called *complete* if it contains three elements, *i.e.* if $c \notin \{AAA, CCC, GGG, TTT\}$. It has been established [29] that there are 20 complete equivalence classes $D_1, \ldots, D_{20}$, each of which encodes three different amino acids or two amino acids and the stop signal. Moreover, it has also been shown [29] that each maximal 1-circular code encoding all 20 amino acids must contain the following seven codons:

$$TGG, ATG, TTC, AAG, GAG, GAC, GGC,$$

which encode the following seven amino acids, respectively:

$$\text{Trp, Met, Phe, Lys, Glu, Asp, Gly,}$$

and belong to the complete equivalence classes $D_2, D_8, D_{11}, D_{15}, D_{18}, D_{19}$.

The basic idea of this proof is to construct the bipartite graph $\mathcal{G} = (V, E)$ displayed in Figure 4.3 where V is the union of the two disjoint sets

$$D = \{D_1, D_3, D_4, D_6, D_7, D_9, D_{10}, D_{12}, D_{13}, D_{14}, D_{16}, D_{17}, D_{20}\}$$

and

$$A = \{\text{Val, Tyr, Thr, Ser, Pro, Leu, Ile, His, Glu, Cys, Asp, Arg, Ala}\}.$$

Moreover, the set of edges E consists of all pairs (D_i, aa) such that there is a codon in D_i that encodes the amino acid aa. It can now be seen that the maximal 1-circular codes that encode all 20 amino acids are in bijective correspondence with the perfect matchings of the constructed graph $\mathcal{G}$. An application of established algorithms for calculating perfect matchings of a graph, *e.g.* the Hungarian algorithm, now yields the list of 52 perfect matchings of $\mathcal{G}$. This completes the proof. □

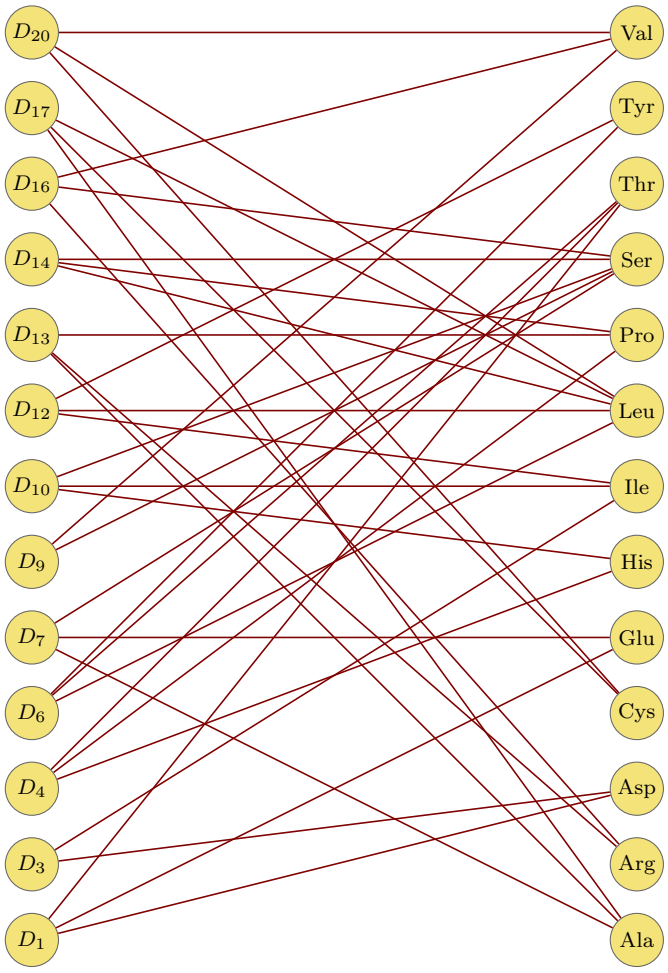


Figure 4.3: The bipartite graph $\mathcal{G} = (V, E)$ where V is composed of the 13 equivalence classes in $D = \{D_1, D_3, D_4, D_6, D_7, D_9, D_{10}, D_{12}, D_{13}, D_{14}, D_{16}, D_{17}, D_{20}\}$ and the 13 amino acids in $A = \{\text{Val, Tyr, Thr, Ser, Pro, Leu, Ile, His, Glu, Cys, Asp, Arg, Ala}\}$, and E is the set of all pairs (D_i, aa) such that there is a codon in D_i that encodes the amino acid aa.

While examining the k-circular codes, a new relation between the *combina-*

torial hierarchy of circular codes and their probability measure of reading-frame retrieval (or reading-frame code in [52]) within two successive codons was discovered (Figure 4.4; for more details of the method see [52]). This reading-frame retrieval measure ranges from 1/3 (one chance out of three to retrieve the reading-frame among the three possible frames in genes) in random codes, *e.g.* $\mathfrak{B}^3$, to 1 (the reading-frame is always retrieved) in strong comma-free codes and comma-free codes. Remember that (i) the maximal size of strong comma-free codes cannot exceed 9 trinucleotides, and there are only 8 codes belonging to this class. Moreover, (ii) there are 408 maximal (size of 20 trinucleotides) comma-free codes. The $12,964,440$ maximal circular codes have reading-frame retrieval values in the range $[72.7, 100]$ (%), including the 216 maximal C^3 self-complementary circular codes in the range $[77.2, 90.1]$ (%). The maximal C^3 self-complementary circular code X identified in genes (see section 1.2.8) has a RFC value equal to 81.3 (%). Noticeably, the identified 52 maximal 1-circular codes have reading-frame retrieval values in the range $[62.2, 71.6]$ (%). Thus, their ability to retrieve the reading-frame is weaker than that of the maximal circular code of the lowest reading-frame retrieval value [72.7] (%). On the other hand, the genetic code $\mathfrak{B}^3$, which is non-circular, has a reading-frame retrieval probability equal to 1/3, as do all random codes (depicted in Figure 4.4). The 4 unitary codes $\{AAA\}$, $\{CCC\}$, $\{GGG\}$ and $\{TTT\}$, which are non-circular, are also random codes with a reading-frame retrieval probability equal to 1/3. It is noteworthy that the loss of the reading-frame in a sequence of such a unitary code does not cause the coding of an amino acid different from the one coded in the reading-frame. In contrast,the 60 remaining unitary codes which are circular and comma-free (48 strong comma-free). In summary, the growing combinatorial hierarchy of k-circular trinucleotide codes is associated with a decreasing probability hierarchy of reading frame coding reading-frame retrieval.

The main property of circular codes, which has first been reported in 1996, is the nucleotide window length for retrieving the reading-frame, *e.g.* 15 nucleotides with the circular code X in genes. The relation identified above shows that, from our point of view, the circular codes may retrieve the reading-frame in genes according to two properties (property (i) is classic in coding theory, property (ii) is a new proposition): (i) always retrieved, *i.e.* without error, using a nucleotide window length but with a slow process; and (ii) retrieved with high frequencies, *i.e.* not always as some errors may occur, within two successive codons, thus with a fast process.

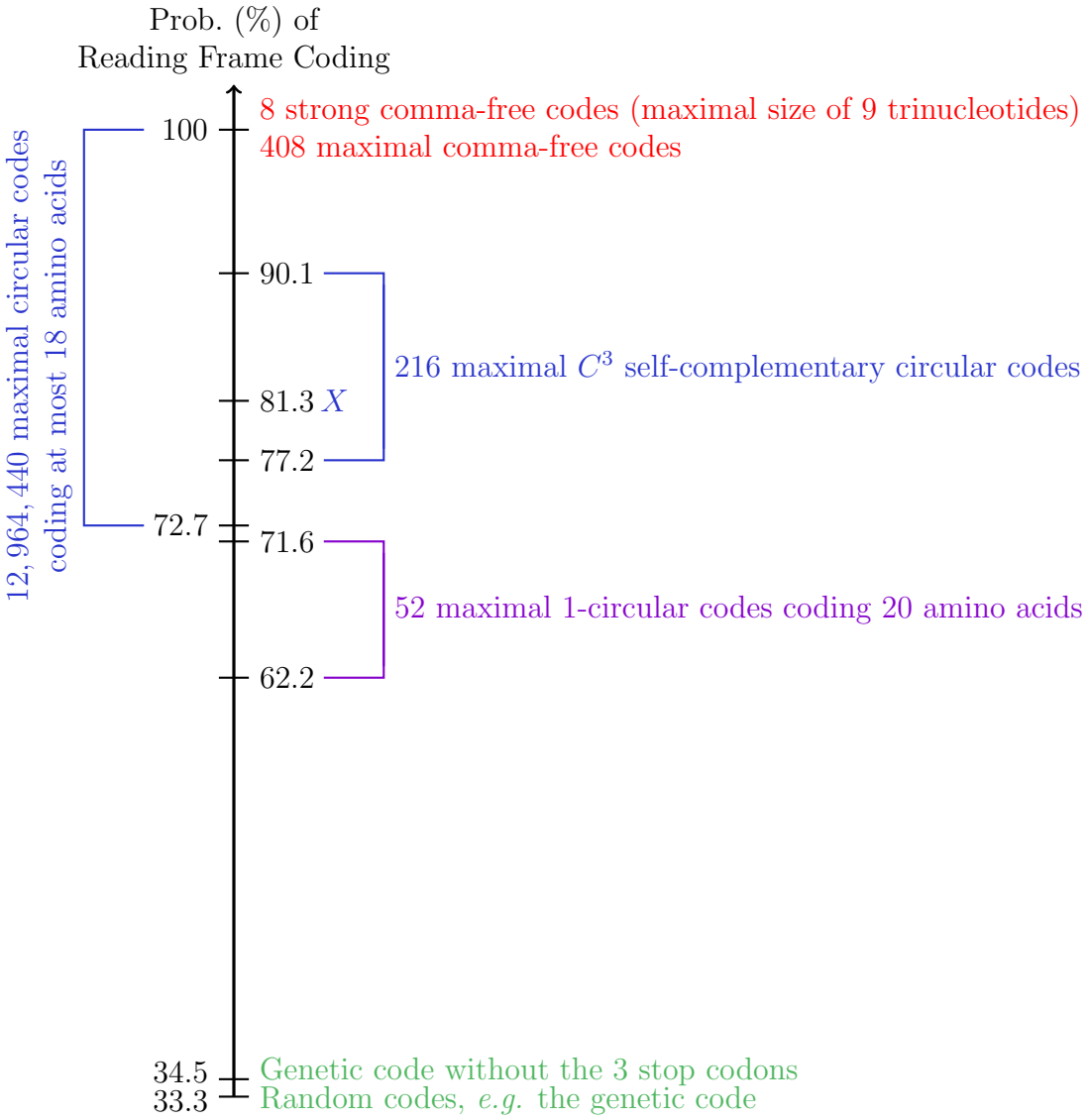


Figure 4.4: A probability hierarchy of reading-frame retrieval within two successive codons with the circular (4-circular) trinucleotide codes and the 52 maximal 1-circular trinucleotide codes coding the 20 amino acids (updated from [52]*Figure 1).

4.5 Summary of Chapter 4

Chapter 4 introduces the class of k-circular codes, a new class of block codes. A k-circular code is defined as a code which requires a concatenation of a maximum of k words written on a circle to reliably ensure the reading-frame. Consequently, $k+1$ words written on a circle could be read in more than one reading-frame.

Therefore, we can extend the definition of k-circular codes by the fact that each of these codes must also be a $k-1$-circular code. It can further be seen that a k-circular circle code can be a $k+1$-circular circle code but does not have to be. The class of k-circular codes contains both circular and comma-free codes as subclasses for every given k.

The main Theorem 4.3.1 of the chapter presents the integer $k(n,\ell)$ where n is the cardinality of the alphabet, and ℓ is the word length of the code. This number $k(n,\ell)$ is the border so that a $k(n,\ell)$-circular code must be a circular code. In Table 4.1, the border values $k(n,\ell)$ for a dinucleotide, a trinucleotide and a tetranucleotide are listed.

Dinucleotide	Trinucleotide	Tetranucleotide
$k(4,2)=4$	$k(4,3)=4$	$k(4,4)=16$

Table 4.1: The $k(n,\ell)$ value for a dinucleotide, a trinucleotide and a tetranucleotide. A $k(n,\ell)$-circular code must be a circular code. For all listed values $n \coloneqq |\mathfrak{B}|$ is four. The value for ℓ is either $2, 3$ or 4

Theorem 4.3.1 has been a major breakthrough. In combination with the Theorem 4.3.2, the decision whether a code is circular or not is no longer decidable only by representing the code in a graph, but also combinatorially in finite time. Before this revelation, a code always had to be transformed into a graph to determine if it is a circular code. With $k(n,\ell)$, a verification of a finite combination can now provide this result.

With the insights from Lemma 4.4.1, it can be proven that 52 of all $3,486,784,401$ 1-circular codes encode all 20 amino acids. Using this knowledge, we were able to identify the 1-circular codes as the only known block codes with a word length $\ell = 3$ over the genetic alphabet to encode all 20 amino acids.

In the final section 4.4, the probability measure of reading-frame retrieval is introduced. This value was originally introduced in [52] and can be used to compare the reliability of block codes in respect to their reading-frame retrieval qualities.

Chapter 5

Codes in Sequences

In this final chapter, the application of theoretical models will be outlined with the aim of connecting the mathematical results from the above chapters with protein synthesis. The methods applied in this chapter are intended to reveal possible evidence for error-correcting codes in biological mRNA sequences. An additional focus lies on the provision of assistance to existing model of the X-code. Recalling the origin of the X-code, it was obtained by observing data from a massive analysis of the genetic information of different species. Such procedures, as they can be found in+ [1], are complex and expensive. However, not only has the hardware become more powerful over the years with increasing computing power, but the algorithms for using these capacities have also become more sophisticated. Hence, we would like to present a software tool and methods that promote the investigation of theories such as those outlined in Chapter 2 and Chapter 4. The aim is to offer theoretical models of a hypothetical reading-frame retrieval machinery in genes. The presented results have only been obtained from coding sequences (CDS) of the species listed in Table 5.1. The obtained results are promising, but the empirical relevance of the values must be increased by more detailed and specific tests done in the future.

In the first section 5.1, we introduce the R package **GCATR**. This project is a further development of **GCAT**. **GCAT** was developed as a stand-alone tool for the examination of circular codes in RNA and DNA sequences [42]. At the beginning of **GCATR**, the tool was intended to be a copy of **GCAT** that can be used in R. However, the tool has grown beyond the original version. It has not only been extended by the algorithms developed in this work, but it also contains new theories and features, such as the theory of conductivity of the genetic code by P. Błażej, D. Kowalski, D. Mackiewicz, M. Wnetrzak, D. Aloqalaa and P. Mackiewicz [7]. The outline of **GCATR** contains a short version of the manual and gives insight into some architectural features [71].

Section 5.2 will proceed with demonstrating two different methods to finding

evidence for error-correcting codes in genes. Both methods are theoretical models of a possible reading-frame detection. While these methods are not directly related to the circular codes, they describe a hypothetical way in which the ribosome might be able to detect reading-frame errors. Although the proof of the actual implementation of these methods in the ribosome is rather demanding, the results of applying them to coding sequences (CDS) are intriguing.

The methods presented in section 5.2 will then be applied to RNA sequences in section 5.3. The outcomes of the methods applied to CDSs will be listed in section 5.2.1. In this context, "to apply" indicates reviewing the coverage of codes using the methods in sequences. More precisely, it indicates reviewing of the coverage of the regions in a sequence where the codes can retrieve the correct reading-frame. The results are optimized codes that have the highest possible coverage based on the methods.

5.1 Development of the research software GCATR for circular codes

GCATR is an open source R package available on Github. While it is still a beta version, it has been downloaded by several users. The tool requires basic knowledge of the computer language R, a so-called interpreted language. However, it is more complex than that as R is an environment with a variety of statistical and graphical techniques [34]. The subsequent chapter will outline the architecture of **GCATR** with its significant features and its shortcomings.

5.1.1 Architecture of the GCATR

One of the shortcomings of R is the interpretation of the language at run-time. While this allows R to be easily installed, learned, and executed, it also increases the execution time. It can be argued that R is optimized for working with large data. However, this has also been achieved by writing bottleneck algorithms in C++. Thus, it was decided that the main business logic of **GCATR** will also be written in C++. The layer architecture of the **GCATR** package is depicted in Figure 5.1.

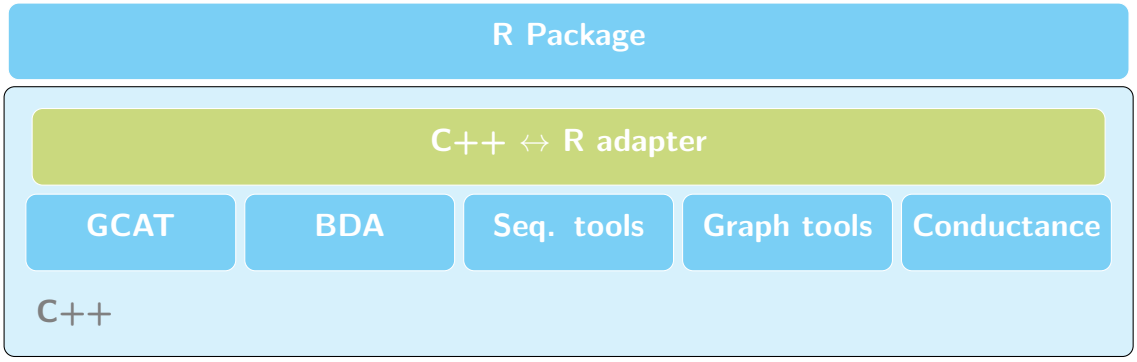


Figure 5.1: This layer diagram shows the layers of the **GCATR** package. The main business logic in the bottom layer is written in C++. This layer was developed in five packages: *GCAT, BDA, Sequence tools, Graph tools* and *Conductance*. An adapter layer, using the package RCPP, allows access to the C++ classes from R. A final upper layer assembles the C++ components and manages their public appearance.

Core architecture of GCATR

The design of the architecture of the **GCATR** C++ part uses the delegation pattern. Figure 5.2 and Figure 5.3 offer a simplified overview of the implemented architecture. The core of the package is the interface *WordContainer*. A *WordContainer* is an object which contains words (tuples of letters) like a sequence or a code. This interface *WordContainer* is implemented by several classes. Each of these classes is equipped with a validation function, which is adapted to the specific needs of each class. For instance, for the function of the gene-related classes, it verifies whether the codes consist only of nucleotides, while for the validation of the Tessera code class, for example, it analyzes whether the words are all correct Tesserae.

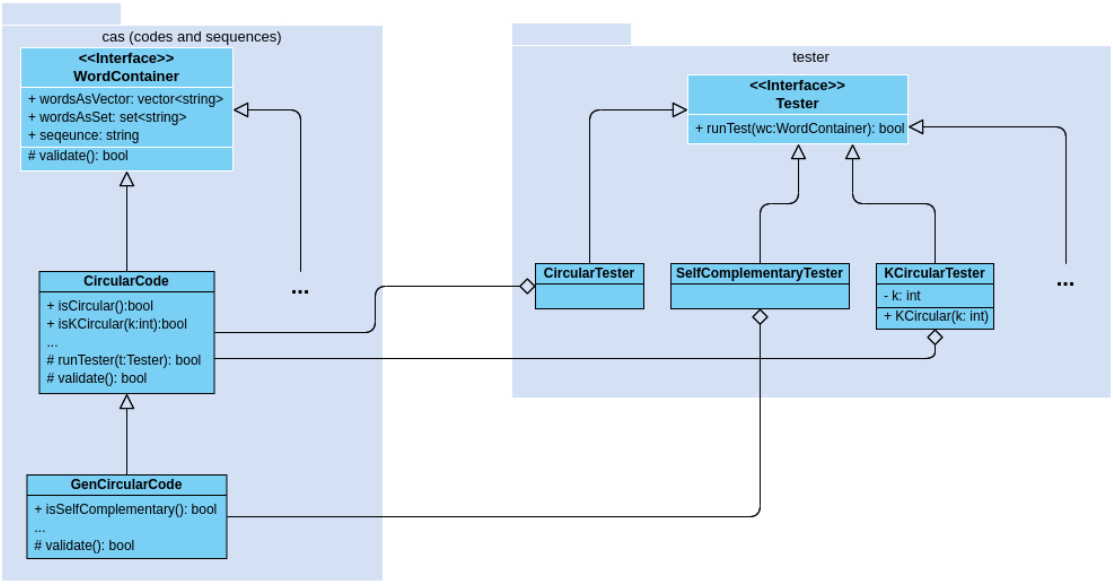


Figure 5.2: Simplified version of the architecture of **GCATR**. The classes in the namespace *cas* delegate property testing to the individual test classes. All test classes extend the interface called Tester. The CircularCode class is only one of many classes that extend the WordContainer interface. Other classes are TesseraCode, GenCircularCode, GenSequeunce, and MixedCircularCode.

The system has three different types of delegates: the tester, the modifier, and the miner. The separation of these three types is due to the return value of the executing function.

tester The delegates grouped as tester return a boolean value (True or False) when executed. They test whether an object that inherits from WordContainer has a certain property, e.g. circularity or freedom from commas.

modifier The group gathered as modifier has no return value when executed. A modifier influences the code/sequence of the delegator, i.e. the WordContainer is changed in some way. These changes can be, for example, transformations or permutations.

miner A miner delegate returns an element. These elements are properties of the code, e.g. the longest path or all cyclic paths in the graph associated with the code. The elements are returned as void pointers (anonymous object) and have to be cast into the expected object by the delegator.

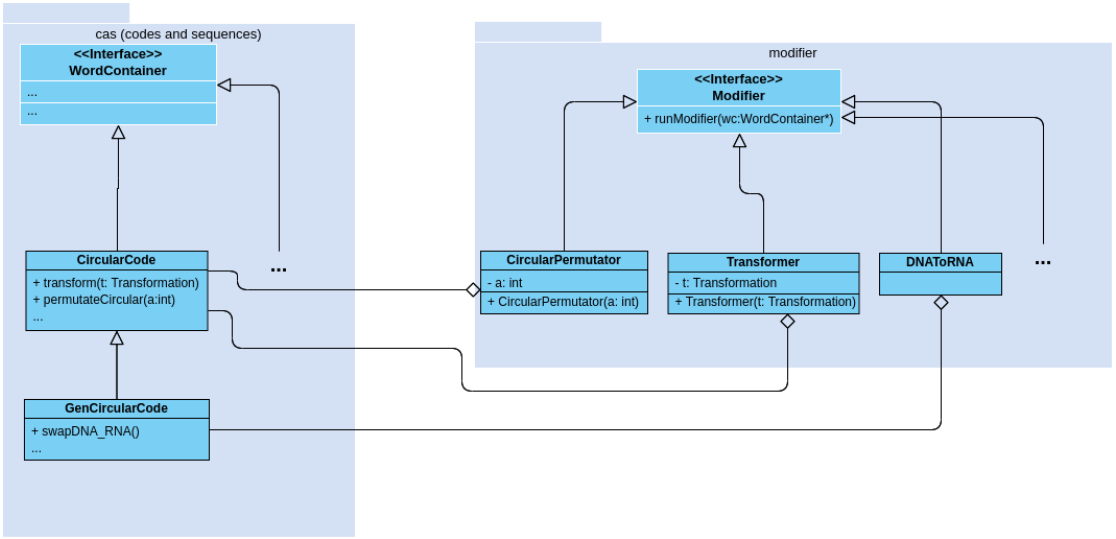


Figure 5.3: Simplified version of the architecture of **GCATR**. The classes in the namespace *cas* delegate the modifications to the individual modifier classes. All modifier classes extend the interface called Modifier.

Package GCAT

The stand-alone tool **GCAT** is written in Java and comes with a comprehensive editor. It allows the user to enter sequences and codes into a user interface. Even though the tool has a great user experience, it has some practical flaws. Installation, for example, is an obstacle that prevents many users from using it. Additionally, the necessary installation of Java is a problem not tolerated by many users. Hence, the design of the new tool was developed to accommodate a large group of users. As many researchers are already using R as a tool for data science, it was a logical decision to restructure the project in R. Subsequently, we list selected examples to illustrate the work with **GCATR**. The entire manual can be found on the web page `https://github.com/StarmanMartin/GCATR`.

The first example returns the maximum number of words of the code $\mathcal{X} = \{ACG, CGT, TCC, CCA\}$, which can be written on a cycle and reliably be decomposed in only one reading-frame into words of $\mathcal{X}$. In this context, it is called the k value (see chapter 4) of a code:

```
1 code_k_value(c("ACG", "CGT", "TCC", "CCA"))
```

R console output:

```
→ 1
```

The same results can be achieved with the following arguments:

```
code_k_value("ACGCGTTCCCCA", 3)
code_k_value("ACG␣CGT␣TCC␣CCA")
```

R console output:

```
→ 1
→ 1
```

As the example outlines, all functions can be called in three different ways. The code can either be passed to the function as an array, as a string separated by white spaces or as a sequence with an additional word length parameter. A factory class then decides whether the code is a code of codons, a Tessera code, any other kind of RNA/DNA code or a non-genetic code.

Other functions, including *code_check_if_comma_free*, *code_check_if_cn_circular*, *code_check_if_code* and *code_check_if_circular*, can be called with the same settings of parameters. Additional functions referring to Chapter 3 have also been implemented, namely *codons_to_tessera* and *code_check_if_tessera*.

Another feature of the package allows for translating codons into amino acids. The following translation tables are available in the tool:

- The Standard Code
- The Vertebrate Mitochondrial Code
- The Yeast Mitochondrial Code
- The Mold, Protozoan, and Coelenterate Mitochondrial Code, and the Mycoplasma/Spiroplasma Code
- The Invertebrate Mitochondrial Code
- The Ciliate, Dasycladacean, and Hexamita Nuclear Code
- The Echinoderm and Flatworm Mitochondrial Code
- The Euplotid Nuclear Code
- The Bacterial, Archaeal, and Plant Plastid Code
- The Alternative Yeast Nuclear Code
- The Ascidian Mitochondrial Code
- The Alternative Flatworm Mitochondrial Code
- The Chlorophycean Mitochondrial Code

- The Trematode Mitochondrial Code
- The Scenedesmus Obliquus Mitochondrial Code
- The Thraustochytrium Mitochondrial Code
- The Pterobranchia Mitochondrial Code
- The Candidate Division SR1 and Gracilibacteria Code
- The Pachysolen Tannophilus Nuclear Code
- The Karyorelict Nuclear Code
- The Condylostoma Nuclear Code
- The Mesodinium Nuclear Code
- The Peritrich Nuclear Code
- The Blastocrithidia Nuclear Code

The list of all tables can be accessed by the function *print_all_translation_tables*. Each table is listed with an index. To obtain the actual codon amino acid mapping, the function *genetic_codes_by_name* or *genetic_codes_by_index* can be used as depicted in the example below:

```
code <- genetic_codes_by_name("The Yeast Mitochondrial Code")
```

or

```
code <- genetic_codes_by_index(1)
```

Two functions are offered to translate a code into amino acids. The first one translates the codons in the same order, while the second one lists a unique set of the encoded amino acids.

```
code_get_all_amino_acids("ACG CAA CAG", idx_trans_table=2)
```

R console output:

```
→ "Thr" "Gln" "Gln"
```

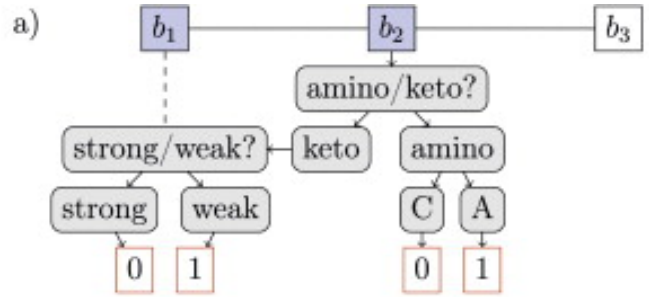
```
code_get_amino_acids("ACG CAA CAG", idx_trans_table=2)
```

R console output:

```
→ "Gln" "Thr"
```

Package Binary dichotomic algorithm (BDA)

a)



b)

	U		C		A		G		
U	Phe	111	Ser	001	Tyr	101	Cys	111	U
U	Phe	111	Ser	001	Tyr	101	Cys	111	C
U	Leu	110	Ser	010	!	110	!	110	A
U	Leu	100	Ser	000	!	100	Trp	100	G
C	Leu	010	Pro	000	His	100	Arg	010	U
C	Leu	010	Pro	000	His	100	Arg	010	C
C	Leu	010	Pro	010	Gln	110	Arg	010	A
C	Leu	000	Pro	000	Gln	100	Arg	000	G
A	Ile	111	Thr	001	Asn	101	Ser	111	U
A	Ile	111	Thr	001	Asn	101	Ser	111	C
A	Ile	110	Thr	010	Lys	110	Arg	110	A
A	Met	100	Thr	000	Lys	100	Arg	100	G
G	Val	011	Ala	001	Asp	101	Gly	011	U
G	Val	011	Ala	001	Asp	101	Gly	011	C
G	Val	011	Ala	011	Glu	111	Gly	011	A
G	Val	001	Ala	001	Glu	101	Gly	001	G

c)

BDA	(i_1, i_2)	Q_1	Q_2
Rumer	(2, 1)	(C, A)≙Am	{C, G}≙ S
Parity	(3, 2)	(G, A)≙ R	{C, A}≙ Am
Complementary	(1, 3)	(C, G)≙ S	{G, A}≙ R

Figure 5.4: This figure (src.: [41]) is a summary of the workflow of the Binary dichotomic algorithms (BDAs). **a)** Displays one representative of the BDAs (Rummer-class algorithm), **b)** is a classification of all codons by the means of three overlapping BDAs, and **c)** lists all parameters for the three BDAs used.

The BDAs were first introduced by Fimmel, Danielli, and Strüngmann [19]. A BDA consists of two so-called questions $Q_1 = (b_1, b_2)$ and $Q_2 = (b_3, b_4)$ and two indices $i_1, i_2 \in \{1, 2, 3\}$ where $i_1 \neq i_2$ as demonstrated in Figure 5.4 **(c)**. A partition of a codon $w = b_1b_2b_3 \in \mathfrak{B}^3$ by the means of the BDA is defined as follows (see Figure 5.4 **(a)**):

(A) if $b_{i_1} \in Q_1$, then
$\rightarrow$ result is 0 if $b_{i_1} = b_1$ or 1 if $b_{i_1} = b_2$

(B) if $b_{i_1} \notin Q_1$, then
$\rightarrow$ result is 0 if $b_{i_2} \in Q_2$ or 1 if $b_{i_2} \notin Q_2$

In **GCATR**, this was implemented as represented in the flowchart in Figure 5.5.

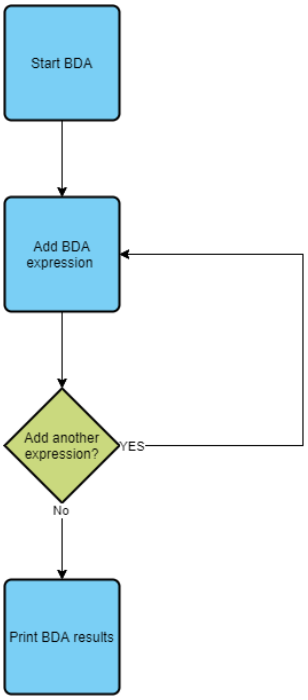


Figure 5.5: This flowchart demonstrates the process of executing a BDA procedure in **GCATR**

```
library(GCATR)

bda <- start_bda()
add_rule(bda, 3, 2, "C", "A", "C", "G")
add_rule(bda, 3, 2, "G", "A", "C", "A")
add_rule(bda, 1, 3, "C", "G", "G", "A")
res <- run_bda_as_matrix(bda)
```

The output is a table of binary triplets representing all trinucleotides. The values are the same as in the table illustrated in Figure 5.4.

Package Sequence tools

The investigation of block codes in sequences is one of the major aspects of **GCATR**. Basic information can be accessed easily , such as the used alphabet, the number of all words in the sequence and even the actual number of times a word occurs in the sequence. All of this is contained in the return value of the function:

```
res <- seq_get_info("ACGCGAACG", 3)
```

R console output:

```
res$alphabet → "TCAG"
res$number_of_tuple → 2
res$tuple_count → ACG: 2 CGA: 1
```

A more complex function is the one called: *find_and_analysis_code_in_sequence*. Apart from other information, this function lists all motifs of a code in a sequence, the position of the motifs in the sequence, the remaining parts which are not motifs and the circular permutation of each word in the sequence in terms of the words in the code.

```
seq <- "ACGTCGCGACGTACGACGTCGTACTCGATGCAAGATC"
res <- find_and_analysis_code_in_sequence(seq, "ACG␣TCG")
```

R console output:

```
res$word → "ACG" "TCG" "ACG" "ACG" "TCG" "TCG"
res$idx_list → 0 3 12 15 18 24
res$rest → "CGACGTTACATGCAAGATC"
res$parts → """ACGTCG""CGACGT""ACGACGTCG""TAC""TCG""ATGCAAGATC"
res$longest_match → 9
res$total_match_in_percent → 48.64865
res$circular_permutations → 1 1 3 3 1 1 1 0 1 0 0 0 0
```

Package Graph tools

A directed graph associated with a code is one of the most vital tools used for the investigation of circular codes. Hence, the **GCATR** has implemented a wide range of functions to work with such a graph. The graph displayed in Figure 5.6 is the result of the example shown below.

```
plot(code_factor_graph("ATT␣CAG␣CTG␣AAG␣TAC␣GGA␣ACG"))
```

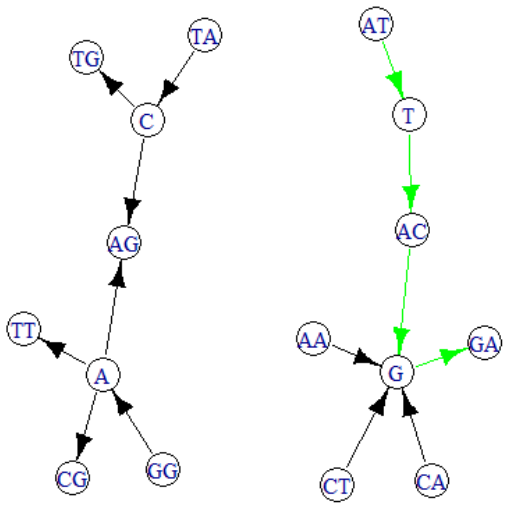


Figure 5.6: The directed graph associated with the code {*ATT, CAG, CTG, AAG, TAC, GGA, ACG*}. The green edges mark the longest path in the graph.

The graph factory function uses the R igraph package `https://igraph.org/r/`. In the graph, the longest paths are marked by green edges and all cycles are marked by red edges. These features of an associated graph can also be displayed separately with the functions: *code_factor_cycle* and *code_factor_longest_path*.

In addition to these two functions, it is also possible to work with the longest path or a cyclic path as sequences or vectors. This can be achieved with the functions *code_get_all_cycles_as_vector* and *code_get_all_longest_path_as_vector*.

Package Conductance

The article "The structure of the genetic code as an optimal graph clustering problem" introduced the value of conductance for the first time[7]. The conductance is a value which gives an intuition of the potential negative impact of a point mutation of the protein syntheses. It has thus been promoted as an evolutionary factor of the standard genetic code[1]. The functionality related to this model is listed below:

- get_average_conductance_of_code returns the average conductance of a code
- get_max_conductance_of_code returns the minimal conductance of a code

[1]For more details see the article [7].

- get_min_conductance_of_code returns the maximal conductance of a code

For more information about **GCATR**, visit the website `https://github.com/StarmanMartin/GCATR` where you can find the source code and detailed information about the functionality.

5.2 Methods

This section contains a listing of the CDS used to gain the results. Furthermore, it explains the two theoretical models of a possible reading-frame detection that have been mentioned in the introduction to this chapter. Subsequently, it introduces a feature of motifs in sequences, which combines k-circularity (see chapter 4) and the results related to the paths in a representing graph (see chapter 2). Afterwards, this chapter guides you through an implementation of the so-called hillclimber algorithm. We close this chapter with an explanation of a quality feature of codes that serves as a measurable property to compare the results.

5.2.1 Coding Sequences (CDS)

The following section lists the used coding sequences. However, due to the fact that the calculation time for this version had to be reduced, some algorithms only used a scaled down version of the listed sequences. The set of sequence selection was taken from [43] after consultation with the author Gumbel, M..

Species	**#CDS**	**#Bases**	**#Codons**
Human	32554	55631471	18543823.6666667
C. elegans	33111	46925943	15641981
Yeast	5917	8899818	2966606
E. coli	5494	4762104	1587368
Herpes virus	118	174237	58079
SARS virus	11	29277	9759

Table 5.1: The used sequences contain the given number of bases and codons.

The coding sequences were downloaded from the resources depicted in the sections below. All sequences use Thymine (T) instead of Uracil (U), even though they represent RNA sequences.

Human

DNA sequences were taken from the Consensus CDS (CCDS) project.
https://www.ncbi.nlm.nih.gov/projects/CCDS/CcdsBrowse.cgi
and downloaded from FTP server:
ftp://ftp.ncbi.nlm.nih.gov/pub/CCDS/

Yeast (Saccharomyces)

https://www.yeastgenome.org(https://www.yeastgenome.org
https://downloads.yeastgenome.org/sequence/S288C_reference/orf_dna/

Worm (C. elegans)

https://www.ebi.ac.uk/ena/data/view/Taxon:Caenorhabditis%20elegans
Then "Coding (Release)":
https://www.ebi.ac.uk/ena/data/view/Taxon:6239&portal=coding_release
Download via "FASTA" link. File name is celegans.fasta.

Bacteria (E. coli)

http://bacteria.ensembl.org/Escherichia_coli/Info/Index
Then "Download DNA sequence (FASTA)".
ftp://ftp.ensemblgenomes.org/pub/bacteria/release-44/fasta/bacteria_91_collection/escherichia_coli/cds/

Viruses

https://www.ebi.ac.uk/genomes/virus.html
Two viruses were selected:

- Abalone herpesvirus Victoria/AUS/2009 or short 'herpes'.
  https://www.ebi.ac.uk/ena/data/view/Taxon:1241371
  Then "Coding (Release)":
  https://www.ebi.ac.uk/ena/data/view/Taxon:1241371&portal=coding_release
  Download via "FASTA" link. File name is ena-herpes.fasta.
- Bat SARS coronavirus HKU3 or short 'sars'.
  https://www.ebi.ac.uk/ena/data/view/Taxon:442736
  Then "Coding (Release)":
  https://www.ebi.ac.uk/ena/data/view/Taxon:442736&portal=coding_release.
  Download via "FASTA" link. File name is ena-sars.fasta.

Scaled down sequence set

The scaled down sequence set includes ten sequences (if available) of each species. These randomly selected sequences have a length of at least 160 nucleotides (if available). All algorithms that are executed with this data are executed multiple times on different, randomly generated, scaled down sequence sets. The results are almost identical as the numbers differ only by an acceptable error ϵ. The error of the percentage coverage of the different methods used is on average $\epsilon < 0.5\%$. The set of coding sequences for the presented numbers in section 5.3 can be found in the Appendix Section II.3.

5.2.2 Random codes and sequences

Random sequences as well as random codes are often valuable tools to verify results. In the subsequent section, we often refer to such random codes and random sequences. The random sequences used in this thesis are uniformly distributed. Any sequence used in the experiments is actually a set of 100 generated sequences of length 1500 nucleotides.

Definition 35. *All random sequences are 100 randomly generated sequences each of length 1500 nucleotides.*

If a sequences is uniformly distributed, a random code is not. It is actually a 1-circular random code. Each code is a set of 20 codons, where each belongs to a different complete circular permutation class (see Definition 3).

Definition 36. *A 1-circular random code is a set of randomly selected codons from each of the 20 complete circular permutation classes.*

5.2.3 Reading-frame retrieval method 1

The first method outlined in this chapter analyzes whether the three words at the same position in the three reading-frames are present in different circular permutations of the same code $\mathcal{X}$. Suppose w_1 is the word in the normal reading-frame, w_2 is the word in the $+1$ reading-frame, and w_3 is the word in the $+2$ reading-frame. Then, the reading-frame is confirmed if $\alpha_{i1}(w_1), \alpha_{i2}(w_2), \alpha_{i3}(w_3) \in \mathcal{X}$ so that $i1 \neq i2 \neq i3$. Figure 5.7 gives an example.

Figure 5.7: This figure gives an example of the first reading-frame retrieval method based on the X-code. It shows that $\alpha_1(TGT)$ is in the X-code, $\alpha_0(GTA)$ is also in the X-code, and $\alpha_2(TAT)$ is in the X-code. Since all three words are different circular permutations of words in the X-code, the reading-frame is confirmed
.

According to the method explained in Figure 5.7, we outline a pseudocode to calculate the coverage of the method in a sequence.

Pseudocode 3. ***Require:*** $\mathcal{X} \subset \mathfrak{B}^3$, $\ell := 3$ *and* $seq \in \mathfrak{B}^+$

function READINGFRAMERETRIEVALMETHOD1$(\mathcal{X}, seq)$

$cp_0 \leftarrow find\_and\_analysis\_code\_in\_sequence(seq, \mathcal{X})\$circular\_permutations$

$cp_1 \leftarrow find\_and\_analysis\_code\_in\_sequence(\alpha_1(seq), \mathcal{X})\$circular\_permutations$

$cp_2 \leftarrow find\_and\_analysis\_code\_in\_sequence(\alpha_2(seq), \mathcal{X})\$circular\_permutations$

$wordsInSeq \leftarrow \lfloor \frac{|seq_0|}{\ell} \rfloor$

$res \leftarrow 0$

for $wIdx \in 0 \ldots wordsInSeq$ ***do***

if $IsUnique(cp_0[wIdx], cp_1[wIdx], cp_2[wIdx])$ ***then***

$res \leftarrow res + 1$

end if

end for

return $\frac{100 \times res}{wordsInSeq}$

end function

The code implements a function to calculate the coverage of a code in a sequence under the restrictions of reading-frame retrieval method 1. In this pseudocode function, the GCATR function find_and_analysis_code_in_sequence (see Section 5.1.1) is used. The function $\alpha_j(\cdot)$ is denoted as the circular permutation function (see Section 1.2.1). The return value of the function is the coverage in percent.

5.2.4 Reading-frame retrieval method 2

The second method examines for each position whether the word in the normal reading-frame is in the code, whether the word at the same position in the +1 reading-frame is in the circular 1-permutation of the code and whether the word at the same position in the +2 reading-frame is in the circular 2-permutation of the code. Figure 5.8 gives an example of that.

Figure 5.8: This figure gives an example of the reading-frame retrieval method 2 described above. This example uses the X-code. As can be seen, TGT is not in the X-code and $\alpha_1(GTA)$ is not in the 1-permutation of the X-code. Nevertheless, $\alpha_2(TAT)$ is contained in the X-code. Hence, the reading-frame is confirmed

.

Next, we outline a pseudocode 4 to implement reading-frame retrieval method 2? as described in Figure 5.7.

Pseudocode 4. ***Require:*** $\mathcal{X} \subset \mathfrak{B}^3$, $\ell := 3$ *and* $seq \in \mathfrak{B}^+$

function READINGFRAMERETRIEVALMETHOD1$(\mathcal{X}, seq)$

$cp_0 \leftarrow find\_and\_analysis\_code\_in\_sequence(seq, \mathcal{X})\$circular\_permutations$

$cp_1 \leftarrow find\_and\_analysis\_code\_in\_sequence(\alpha_1(seq), \mathcal{X})\$circular\_permutations$

$cp_2 \leftarrow find\_and\_analysis\_code\_in\_sequence(\alpha_2(seq), \mathcal{X})\$circular\_permutations$

$wordsInSeq \leftarrow \lfloor \frac{|seq_0|}{\ell} \rfloor$

$res \leftarrow 0$

for $wIdx \in 0 \ldots wordsInSeq$ ***do***

if $cp_0[wIdx] = 1$ *or* $cp_1[wIdx] = 2$ *or* $cp_2[wIdx] = 3$ ***then***

$res \leftarrow res + 1$

end if

end for

return $\frac{100 \times res}{wordsInSeq}$

end function

The Code implements a function for calculating the coverage of a code in a sequence under the restrictions of reading-frame retrieval method 2. In this pseudocode function, the GCATR function find_and_analysis_code_in_sequence (see Section 5.1.1) is used. The function $\alpha_j(\cdot)$ is denoted as the circular permutation function (see Section 1.2.1). The return value of the function is the coverage in percent.

5.2.5 Pathless motifs in sequences

This section focuses on the elaboration of a comprehensive feature of code motifs in sequences. We combine the theory of k-circular code (see chapter 4) with the reading-frame results of the longest path theory (see chapter 2).

A k-circular code recognizes a frameshift for any sequence consisting of no more than k words from the code written in one cycle. However, observation 2.1.2

shows that a motif is robust to frameshifting if it is not equal to a word that is a concatenation of the vertices of a path in the graph associated with the code (see Definition 13). In this case, the length of the motif is irrelevant. These motifs are called *pathless motifs*. We must mention that a word that is a concatenation of the vertices of the path can be frameshift stable. This is the case if the path cannot be extended by edges of the associated graph. The method we use to gain results will ignore this fact. Consequently, the actual number of frameshift robust motifs is presumably much higher than the featured results.

5.2.6 Code optimization with a hillclimber algorithm

The hillclimber algorithm was used to optimize the codes with respect to the retrieval of reading-frames. In this section, this algorithm will be elaborated and the way in which it was adapted for our needs will be outlined.

The hillclimber algorithm is a simple but powerful heuristic optimization method. It is an analogy to a mountaineer who is trying to reach the summit in dense fog and therefore steers the steps as steeply as possible uphill. When the only possible steps left are going downwards, the hillclimber has reached a summit. In concordance with the analogy, the hillclimber works according to the following principle: You take a step in a random direction and after each step, the algorithm checks if you are higher than before. If you are, the next iteration starts. If not, you go back and repeat the previous step. The algorithm terminates when it is only possible to go back down.

In order to adapt the code for our usage, we will use the coverage of reading-frame retrieval method 1 and reading-frame retrieval method 2 as fitness values.

First, we will start the algorithm with randomly generated 1-circular codes (see 36). These codes are generated such that they contain a randomly selected codon from each circular permutation class. In a subsequent step, we start the algorithm with the X-code.

The algorithm selects a randomly chosen codon from the code and circularly permutes it at random. Then, it checks whether the fitness has increased. If so, the algorithm starts a new iteration step. If not, the previous step is undone and a new iteration step starts. The algorithm terminates when all possible circular permutations of all codons in the code do not further increase the fitness.

We decided to use this optimization algorithm for two reasons. First, it perfectly fits to an application with many variables and can therefore be easily adapted to the environment. Secondly, to the best of our knowledge, this algorithm is close to the evolutionary optimization process. This argument excludes all external factors of evolution and concerns about the trial and error nature of evolution. Summarized according to Darwin: "Survival of the Fittest" [14].

5.2.7 Code coverage of all reading-frames as quality feature

This section presents a quality feature inspired by the properties of the X-code. First, we would like to recall the X-code discovered by Arquès and Michel [1].

$$\{AAC, AAT, ACC, ATC, ATT, CAG, CTC, CTG, GAA, GAC, GAG, GAT, \\ GCC, GGC, GGT, GTA, GTC, GTT, TAC, TTC\}$$

The X-code is a self-complementary C^3 code, and it was found in sequences of different species. It is a combination of the most common codons in the three reading-frames. To identify the X-code, the most frequent codons in the normal reading-frame are combined with the circular 2-permutation of the most frequent codons in the +1 reading-frame and the circular 1-permutation of the most frequent codons in the +2 reading-frame (see [1]).

As a reference value, we use the percentage of codons in the X-code in the reduced set of sequences from section 5.2.1.

X-code in Normal frame	$\alpha_1(X$-code) in +1 frame	$\alpha_2(X$-code) in +2 frame
41.3%	37.1%	37.7%

On average, the randomly generated 1-circular codes (see Definition 36) of length 20 reveal the following values:

R code in Normal frame	$\alpha_1(R)$ in +1 frame	$\alpha_2(R)$ in +2 rame
≈30.5%	≈30.5%	≈30.5%

The randomly generated codes are generated so that they contain a randomly selected codon from each circular permutation class. The presented values are calculated as an average of 50 random 1-circular codes. The range for all three frames was 25% to 34%.

In the following subsections, we present the results of the two new reading-frame retrieval methods.

5.3 Results

All results presented in this section are those of the algorithms illustrated in section 5.2. These results will strengthen the theory of circular codes in genetic

sequences. This section contains the outcome of the algorithms which (a) provide new hypothetical methods in the ribosome to detect a reading-frame shift and (b) present evidence for the significance of the X-code [1]. In the subsequent section, various circular codes in sequences are discovered. As specified in sections 5.2.6, we will use a hillclimber algorithm. Such an algorithm requires a fitness value to indicate whether it has improved or not. Hence, we use the coverage of a code in terms of the methods elaborated in section 5.2, i.e. the number of positions in the sequence of the code from which the method was able to retrieve the reading-frame.

Before proceeding with the next section, we want to delineate the coverage results of the X-code by applying method 1 (see section 5.2.3) and method 2 (see section 5.2.4), first, to the reduced set of sequences from section 5.2.1, then to randomly generated sequences (see Definition 35).

The results of the X-code and the reduced set of sequences from section 5.2.1:

Method 1 coverage: 51.0%

Method 2 coverage: 64.3%

Average results of the code and randomly generated sequences:

Method 1 coverage: $\approx$ 48%

Method 2 coverage: $\approx$ 52%

5.3.1 Hillclimber with the reading-frame retrieval method 1

In this section, we use the coverage of reading-frame retrieval method 1 (see Pseudocode 3) as fitness value for the hillclimber algorithm introduced in section 5.2.6. As data, the reduced set of sequences from section 5.2.1 is used. We start the algorithm with 100 1-circular random codes to avoid local minimums. The most valuable result is:

$$\mathcal{X}_{resR1} = \{CAA,\ ATA,\ CAC,\ CAT,\ TTA,\ CAG,\ CTC,\ CTG,\ AAG,\ CGA,\ GAG,\ ATG,\ CGC,\ CGG,\ GTG,\ TAG,\ CGT,\ TTG,\ CTA,\ CTT\}$$

Method 1 coverage: 63.0%

The code $\mathcal{X}_{resR1}$ is a non-self-complementary C^3 code. The longest path in the associated graph is of length six. According to Theorem 2.1.4, the reading-frame number is $n_{\mathcal{X}} = 12$. The code $\mathcal{X}_{resR1}$ encodes nine amino acids and the stop signal: *Arg, Gln, Glu, His, Ile, Leu, Lys, Met, Val* and *Stop*. The coverage for method 2 is 49.8%.

$\mathcal{X}_{resR1}$ code in Normal frame	$\alpha_1(\mathcal{X}_{resR1})$ in +1 frame	$\alpha_2(\mathcal{X}_{resR1})$ in +2 frame
31.1%	32.2%	31.4%

Next, we start the algorithm with the X-code. Let us assume we are in the middle of an evolutionary process where the X-code is a relevant factor for this development. In the next step of this hypothetical process, we attempt to simulate the future evolution. In this experiment, we assume that the evolution works according to the hillclimber algorithm, i.e. based on a trial-and-error system. In such a situation, we must assume that the correct course of development is heading for the next local maximum. The result of the hillclimber with the X-code is:

$$\mathcal{X}_{resX1} = \{AAC,\ AAT,\ ACC,\ ATC,\ ATT,\ GCA,\ TCC,\ GCT,\ GAA,\ GAC,\ GGA,\ GAT,\ GCC,\ GGC,\ GGT,\ GTA,\ GTC,\ GTT,\ TAC,\ TTC\}$$

Method 1 coverage: 62.1%

The code $\mathcal{X}_{resX1}$ is a C^3 code. It is non-self-complementary. The longest path in the associated graph is of length six. According to Theorem 2.1.4, the reading-frame number is $n_{\mathcal{X}} = 13$. The code $\mathcal{X}_{resX1}$ encodes eleven amino acids: *Ala, Asn, Asp, Glu, Gly, Ile, Phe, Ser, Thr, Tyr* and *Val*. The coverage for method 2 is 42.9%.

$\mathcal{X}_{resX1}$ code in Normal frame	$\alpha_1(\mathcal{X}_{resX1})$ in +1 frame	$\alpha_2(\mathcal{X}_{resX1})$ in +2 frame
21.7%	23.9%	23.0%

5.3.2 Hillclimber with the reading-frame retrieval method 2

Now, the coverage of reading-frame retrieval method 2 (see section 5.2.4) is used as fitness. As above, we use the reduced set of sequences from section 5.2.1. Again, we start the algorithm with 100 1-circular random codes called R to avoid local minimums. The best results are:

$$\mathcal{X}_{resR2} = \{CAA,\ ATA,\ ACC,\ ATC,\ ATT,\ CAG,\ CTC,\ GCT,\ GAA,\ GAC,\ GAG,\ GAT,\ CCG,\ GGC,\ GTG,\ GTA,\ CGT,\ TTG,\ TAC,\ TCT\}$$

Method 2 coverage: 71.1%

The code $\mathcal{X}_{resR2}$ is neither self-complementary nor circular. However, the code $\mathcal{X}_{resR2}$ encodes 13 amino acids: *Ala, Arg, Asp, Gln, Glu, Gly, Ile, Leu, Pro, Ser, Thr, Tyr* and *Val*. The coverage for method 1 is 40.8%.

$\mathcal{X}_{resR2}$ code in Normal frame	$\alpha_1(\mathcal{X}_{resR2})$ in +1 frame	$\alpha_2(\mathcal{X}_{resR2})$ in +2 frame
38.2%	37.3%	36.4%

Next, we start the algorithm with the X-code as a hypothetical simulation of the evolution. To identify the local maximum that the X-code is heading to in the hypothetical case where method 2 has a major influence on the evolution, the result is:

$$\mathcal{X}_{resX2} = \{CAA,\ ATA,\ ACC,\ ATC,\ ATT,\ CAG,\ CCT,\ GCT,\ GAA,\ GAC,\ GAG,\ GAT,\ CCG,\ GGC,\ GTG,\ GTA,\ CGT,\ TTG,\ TAC,\ TTC\}$$

Method 2 coverage: 71.0%

The code $\mathcal{X}_{resX2}$ is neither self-complementary nor circular. However, the code $\mathcal{X}_{resX2}$ encodes 13 amino acids: *Ala, Arg, Asp, Gln, Glu, Gly, Ile, Leu, Pro, Ser, Thr, Tyr* and *Val*. The coverage for method 1 is 43.3%.

$\mathcal{X}_{resX2}$ code in Normal frame	$\alpha_1(\mathcal{X}_{resX2})$ in +1 frame	$\alpha_2(\mathcal{X}_{resX2})$ in +2 frame
38.0%	38.5%	36.8%

5.3.3 Hillclimber with pathless motifs in sequence

The results in this section refer to the pathless motifs introduced in section 5.2.5. First, we examine the coverage of the sequences by such pathless motifs from the X-code.

Random sequences: ≈10%

Sequences from section 5.2.1: 20.6%

Recalling that these are not the only frameshift robust motifs, the result is promising. It can be seen that at least 50% of the X-motifs are completely frameshift stable. Moreover, there is a considerable gap between the result of the random sequences and the result of the CDS.

Next, we use the coverage of the CDS by such pathless motifs as the fitness for the hillclimber algorithm and highlight two results:

Code $\mathcal{X}_{resRPath}$ is the most valuable result achieved with the hillclimber algorithm. The coverage of 22.9% of CDS by pathless motifs was unparalleled. The data obtained shows that all other coverage values for the other methods are well above average. The code is:

$$\mathcal{X}_{resRPath} = \{CAA,\ AAT,\ CCA,\ CAT,\ ATT,\ GCA,\ CCT,\ GCT,\ GAA,\ GAC,\ GGA,\ GAT,\ GCC,\ GGC,\ GGT,\ GTA,\ CGT,\ GTT,\ ACT,\ CTT\}$$

Pathless motifs: 23.4%

This code $\mathcal{X}_{resRPath}$ is a circular code, but it is non-self-complementary. The longest path in the associated graph is of length six. According to Theorem 2.1.4, the reading-frame number is $n_{\mathcal{X}} = 13$. The code $\mathcal{X}_{resRPath}$ encodes 13 amino acids: *Ala, Arg, Asn, Asp, Gln, Glu, Gly, His, Ile, Leu, Pro, Thr* and *Val*.

Method 1: 62.0%

Method 2: 53.3%

$\mathcal{X}_{resRPath}$ code in Normal frame	$\alpha_1(\mathcal{X}_{resRPath})$ in +1 frame	$\alpha_2(\mathcal{X}_{resRPath})$ in +2 frame
38.5%	37.2%	36.8%

As for method 1 and 2, we attempt to simulate the evolution by starting the algorithm with the X-code. The goal is to identify the local maximum of the X-code if the pathless motifs had a major influence on evolution. The result is:

$$\mathcal{X}_{resXPath} = \{CAA,\ AAT,\ ACC,\ CAT,\ ATT,\ GCA,\ TCC,\ GCT,\ GAA,\ GAC,\ GGA,\ GAT,\ GCC,\ GGC,\ GGT,\ GTA,\ CGT,\ GTT,\ ACT,\ CTT\}$$

Pathless motifs: 23.4%

This code $\mathcal{X}_{resXPath}$ is a circular code, but it is non-self-complementary. The longest path in the associated graph is of length seven. According to Theorem 2.1.4, the reading-frame number is $n_{\mathcal{X}} = 14$. The code $\mathcal{X}_{resRPath}$ encodes 13 amino acids: *Ala, Arg, Asn, Asp, Gln, Glu, Gly, His, Ile, Leu, Ser, Thr* and *Val*.

Method 1 coverage: 59.1%

Method 2 coverage: 55.3%

$\mathcal{X}_{resXPath}$ code in Normal frame	$\alpha_1(\mathcal{X}_{resXPath})$ in +1 frame	$\alpha_2(\mathcal{X}_{resXPath})$ in +2 frame
39.2%	37.0%	37.2%

5.4 Consequences of Chapter 5

First, we recall the coverage results of the X-code:

Method 1 coverage: 51.0%

Method 2 coverage: 64.3%

Pathless motifs: 20.6%

X-code in Normal frame	$\alpha_1(X\text{-code})$ in +1 frame	$\alpha_2(X\text{-code})$ in +2 frame
41.3%	37.1%	37.7%

If we compare the results of the codes found in section 5.3.1 and 5.3.2 with the results of the X-code, on average, the X-code is the best code. The coverage of the X-code is not exceeded by any of the other codes. It can be seen that the results of either method 1 *or* method 2 are higher for the codes found by the hillclimber algorithm but never of both methods. However, it should not be neglected that these codes have been optimized with regard to one of these values. In summary, it shows that these methods have the potential to be of importance for reading-frame retrieval - in particular method 1. Astonishingly, the results indicate that the coverage of method 1 and circularity of the codes are connected. It is even more surprising that the codes are C^3 codes. Note that there is no natural link between the method and C^3 property of codes. Another exceptionalism of method 1 is that the codes do not need to have high codon coverage in the three reading-frames of a sequence and still perform very well. Method 2 achieves remarkable values over 70%. It must also be considered that the codes encoded 13 amino acids. Hypothetically, this method could perform very well as a machinery to synchronize the reading-frame in genes. Nonetheless, it is unclear how reliable these methods are. For instance, the numbers of method 1 reveal that the coverage in random sequences is only slightly below the coverage in gene sequences with respect to the X-code. Future investigation of these methods is needed to determine whether they are a potential role model for new types of block codes and/or as reading-frame retrieval methods in genes.

Finally, we summarize the two results obtained by means of the *pathless motifs*. The first results indicate that the coverage of the three reading-frames as well as the coverage of method 1 and method 2 are somehow linked to the *pathless motifs*. These motifs have an obvious link to the circularity. Hence, it is unsurprising that the codes found by the hillclimber are circular or even C^3. Yet, the links to method 1 and method 2 cannot be reconstructed.

Chapter 6

Conclusion

This concluding chapter is divided into two sections. The first part uses the results obtained in Chapters 2-5 to describe a hypothetical evolutionary scenario based on a *combinatorial hierarchy*. Subsequently, this dissertation closes with a discussion of the achieved results and an outlook on possible future research.

6.1 Biological consequences

Almost every organism living today stores its genetic material in DNA and uses the standard genetic code (SGC) to translate the 64 trinucleotides (codons) into 20 amino acids and the stop signal during protein synthesis. In section 1.1.3 of the introduction, we have presented the three most widely accepted hypotheses about the origin of the genetic code (see for an overview [46, 47]). Let us recall these hypotheses briefly: (1) the stereochemical theory based on the stereochemical attraction between amino acids and specific anti-codons [64, 79], (2) the adaptive theory, which suggests that the genetic code is the result of an optimisation process to be as robust as possible against mutations [76, 35], and (3) the theory of coevolution of the genetic code with amino acid biosynthetic pathways [77].

In the following we will use the theory of k-circular codes and the circular code classes X_1 to X_8 introduced in section 2.1.2 to develop a hypothetical model of the evolution of the genetic code adapted to the limitations of the adaptive theory. The different classes of circular codes can be perfectly integrated into the chain of argumentation of Novozhilov, Wolf and Koonin from [61], which suggests that the SGC is not completely error-tolerant. The three authors claim in [61] that there must be a compromise between the increasing robustness against translation errors and the extension of the coding table. Certainly, the efficiency of the code is another factor that should be added to this line of argument. By extending the genetic alphabet or the word length, it was possible that the code would contain

more information in addition to being error robust (see section 2.2.1). However, such an evolutionary scenario is only conceivable in a hypothetical primitive genetic code, which in this work is represented in the form of Tessera codes. Nevertheless, it can be assumed that even in this hypothetical scenario, the effectiveness and above all efficiency of protein synthesis forced evolution to develop a trinucleotide-based genetic code or translation mechanism at an early stage.

These considerations inspire the construction of a combinatorial hierarchy of genetic codes (error-detecting codes as subcodes of the genetic code) presented in the following, assuming that the development of the genetic code was accompanied by the development of error detection codes and that the genetic code itself had to be extended in the course of evolution. The corresponding evolutionary model starts with circular Tessera codes as a form of a possible primordial code and is then further developed on the basis of the properties of the circular trinucleotide code classes X_1 to X_8 developed in this work and the (k)-circular code classes (see theorem 4.3.1 with $n = 4$ and $\ell = 3$, and Observation 4.3.2).

Figure 6.1 visualises such a combinatorial hierarchy: the evolutionary template shown has two optional starting points (A) and (B). Starting point (A) in Figure 6.1 refers to the self-complementary Tessera code $X_T = \{AACC,\ GGTT,\ GGCC,\ AATT,\ AGTC,\ GACT,\ GATC,\ AGCT\}$, which is self-complementary strong comma-free and can be mapped to a strong comma-free trinucleotide code, which is a subcode of the RNY code (see section 1.2.8). In contrast, the starting point (B) focuses on the maximum circular Tessera codes. There is much evidence that the maximal circular Tessera codes can be mapped onto comma-free codes (see Observation 3.3.1 and Observation 3.4.3), while there is convincing evidence that there is no correlation between maximal circular Tessera codes and strong comma-free trinucleotide codes. Therefore, the evolutionary hierarchy in Figure 6.1 is based on two opposing starting points. From these points on, the combinatorial hierarchy develops with increasing complexity according to the code classes X_p (see section 2.1.2), where p refers to the maximum path length p (from 1 to 8) of their associated graphs $G(X_p)$. Since the maximum path length p refers to the reading frame number $n_{\mathcal{X}}$ defined in Theorem 2.1.4, the circular codes in X_1 are strong comma-free and the codes in X_2 are comma-free and therefore more restricted than the codes in X_3 to X_8. Table 2.7 lists the reading frame number for each class.

Longest path classes:	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Readingframe number $n_{\mathcal{X}}$:	5	6,7	8	9,10	11	12,13	14	15

Copy of Table 2.7. The readingframe number $n_{\mathcal{X}}$ for the eight classes of circular trinucleotide codes.

The maximum self-complementary C^3 code X discovered by Arquès and Michel [1] is the most important representative of the 216 maximum C^3 self-complementary trinucleotide circuit codes introduced in section 1.2.8 and belongs to class X_8. Through the work of Arquès and Michel there is here for the first time a statistical proof of the existence of such a code in the genetic information and its biological role.

After the X_8 code, a circular (4-circular) code with few restrictions, the development could have continued through the even less restrictive classes of k-circular codes (where $k \in \{1, 2, 3\}$). It is assumed that during this hypothetical course of development, the k-circular codes provide a possible transition from circular codes to the current SGC, in which the property of circularity is completely lost (Figure 6.1). This development led to a weakening of the genetic code with regard to its error robustness. One possible theory is that, as compensation, the motifs of a circular code, especially those of the X-code, which are found representatively in the genes of most organisms [55, 15], are used to retrieve reading-frames.

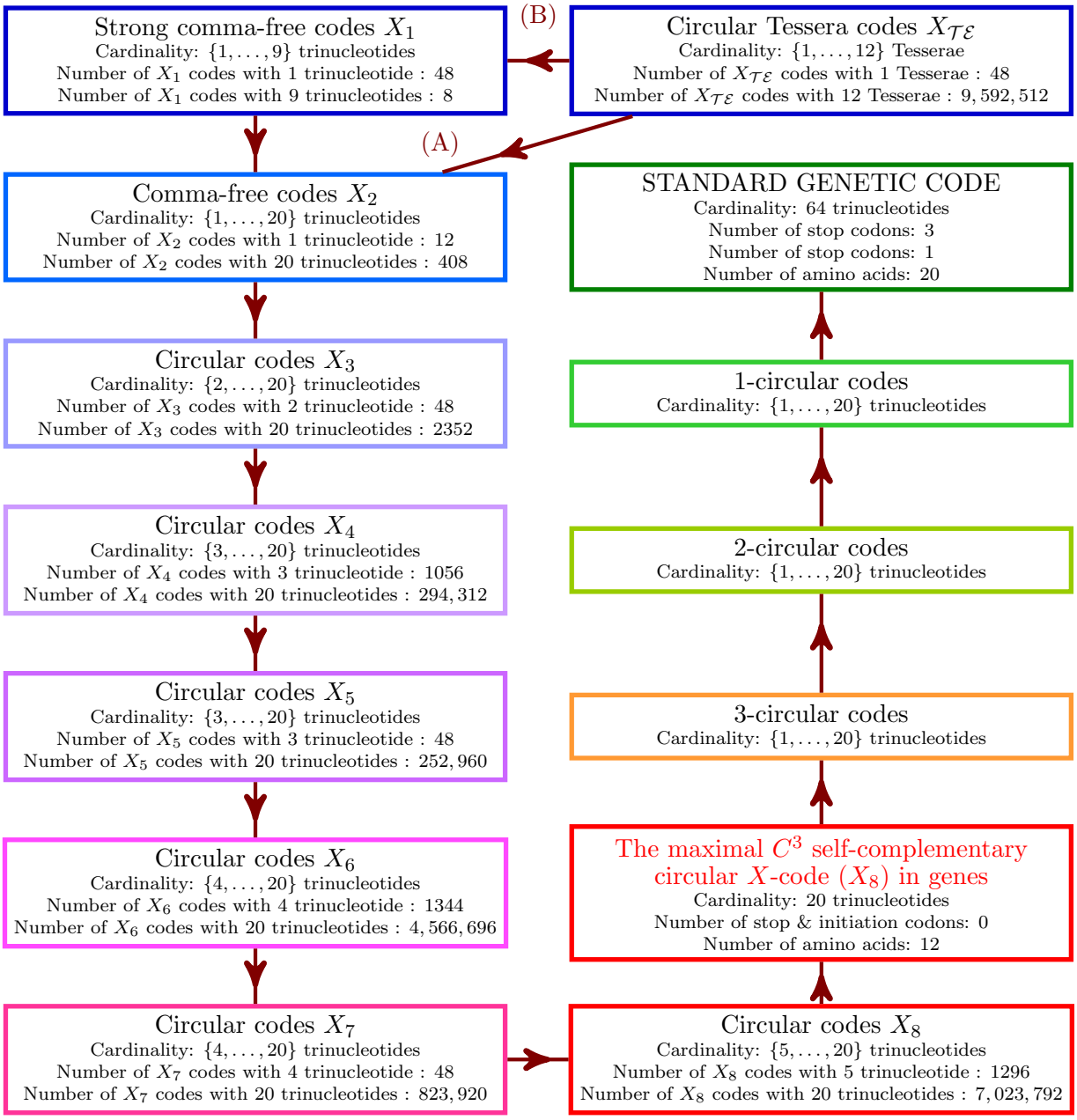


Figure 6.1: A combinatorial hierarchy of circular codes with two potential starting points, the maximal circular Tessera codes with (B), and the strong comma-free Tessera codes with (A). It progresses from circular (4-circular) trinucleotide codes $\mathcal{X}_p$, where p is the maximum path length associated with the graph $\mathcal{G}(\mathcal{X}_p)$, to k-circular trinucleotide codes, where $k \in \{1,2,3\}$ (see Theorem 4.3.1 with $n = 4$ and $\ell = 3$, and Observation 4.3.2).

This evolutionary scenario in the form of a combinatorial hierarchy ultimately provides the SGC of today. As shown in Figure 4.4, the correct reading-frame retrieval of the SGC is not guaranteed. To solve this problem, the authors in [1, 53] suggest that a mechanism synchronises the reading frame that uses motifs of the X-code. In addition, we propose a new hypothetical procedure based on the results of Chapter 5 which presents new ideas on different methods for correctly retrieving the reading frame. Despite their hypothetical nature, the methods in Chapter 5 propose syntax-based reading-fame retrieval systems that are conceivable in a biological mechanism. The results of the developed methods are already very promising and it is obvious that by refining the methods again a novel model of a syntax-based reading-frame retrieval can be developed. However, it must be noted that the possible implementation of such a system in the ribosome has not yet been researched or even observed to the best of the author's knowledge, and thus offers potential for future experimental research. researched or even observed.

6.2 Discussion

The results achieved in this dissertation are discussed from two different perspectives. Both the mathematical and the biological perspective lead to different interpretations of the results. First we will take the biological perspective and it is inevitable to mention that all the theories developed and presented are of hypothetical nature. This is due to the fact that understanding the data is not sufficient to definitively prove and demonstrate the role of circular codes in DNA or RNA sequences, for example. However, this considerably increases the importance of this work, as it serves as a guide to support data collection and to guide future scientific work. As often cited, the X-code is still the strongest and most reliable evidence of error-correcting block codes in protein synthesis. This has been shown in numerous publications starting with[1]. However, the theories about a potential role of comma-free codes in genetic data processing should not be neglected. These theories have led to impressive evidence for the early (in terms of evolution) use of comma-free codes in gene [47]. A perfect link between these two approaches is the classification of circular codes by their longest path length in the associated graphs. This relationship is shown in a hypothetical evolutionary scenario based on a combinatorial hierarchy as shown in Figure 6.1.

This classification is an essential result of the present dissertation. It is not only the quality of correct reading-frame recognition, but also the number of all classes that need to be distinguished. It can be argued that there is currently no evidence for eight evolutionary steps between a strong comma-free codes and the X-code in this hypothetical scenario. However, as both theories are hypothetical, the discovery of such codes in conserved genes would support the model in Figure 4.4

and could lead to a reorientation of evolutionary research.

In this hypothetical evolutionary scenario, which is based on the combinatorial hierarchy, the evolution from the X-code to the standard genetic code (SGC) can ideally be explained by the k-circular codes. This code class adds exactly three hypothetical evolutionary stages to the combinatorial hierarchy. The exact number of steps can be predicted by the sharp boundary of $k(n, \ell)$, from which the $k(n, \ell)$-circularity of an ℓ-letter code $\mathcal{X}$ over a given finite alphabet Σ implies its circularity. Possible reasons against such a development have to face the fact that 1-circular codes are in the remarkable position of being the only known block codes with a word length $\ell = 3$ above the genetic alphabet to code all 20 amino acids.

The theory of circular Tessera codes presented in Chapter 3 led to fundamental results indicating a relationship between self-complementary, strongly comma-free Tessera codes and strongly comma-free trinucleotide codes, and maximal circular Tessera codes and comma-free trinucleotide codes. Although such a connection is as hypothetical as the existence of a possible Tessera code as a predecessor of the present SGC, it is too concise to be neglected.

All the theories developed in this work offer versatile starting points for future biological research. The first proposal refers to Chapter 5 and focuses especially on the comprehensive search for a new method of reading-frame retrieval, which uses the basic ideas of the two reading-frame retrieval methods presented in Chapter 5. A promising research topic is the classification of genes by means of the circular code classes, which are listed in the combinatorial hierarchy in Figure 4.4. For example, motifs as functionally conserved ("ancestral") genes may contain more (k-)circular code motifs compared to functionally specific genes. In addition, code motifs of different circular code classes can be examined with regard to their encoded amino acids.

Finally, we take the (bio-)mathematical perspective. The complete classification of graphs associated with self-complementary circular Tessera codes and self-complementary circular trinucleotide codes of a size ≥ 18 can be very useful for the investigation of such codes. However, this proof still needs to be implemented generally for each word length ℓ. The same applies to the reading-frame number $n_{\mathcal{X}}$ and the classification according to the longest path in the associated graphs. Here a general definition is mandatory and has yet to be found.

The functions dealing with the maximum size of a circular code seem to be finished from today's point of view. In contrast, the mapping function of a circular code to a binary code opens new doors and allows a variety of new approaches, such as those mentioned in section 2.2.3.

In Chapter 3, Table 3.11 opens questions about its properties that are still open to be proven. In addition, we have presented observations on circular trinucleotide

mappings of circular Tessera codes. This relationship has yet to be defined by means of the tubular representation of Tessera codes. The construction of Tessera codes is interesting and complex. However, it still needs to be formalised so that it can be adapted to circular codes of any word length over any finite alphabet. Therefore, a feature of a code has to be linked to properties in the corresponding graph, so that a probability based on acyclic graph components can be found with which this feature occurs. One approach could be the comma-free separation of trinucleotide codes, as explained in section 2.2.2.

The influence of the sharp boundary $k(n, \ell)$ on the study of circular codes is groundbreaking. Such an integer allows to determine whether a code is circular or not. Before the introduction of the $k(n, \ell)$ boundary, the problem could only be solved by first presenting the code as a directed graph and then examining it for its cyclic paths. This discovery will bring new impulses to data science as it significantly simplifies algorithmic complexity. Due to the faster recognition of a circular code, the analysis of large DNA sequences can lead to new results without the need for a "supercomputer".

Chapter I

Definitions and notations

Code theory

$\Sigma \rightarrow$ Denotes an arbitrary finite alphabet. Σ is a set of letters.

$n \rightarrow$ Denotes the cardinality of a alphabet Σ. $n := |\Sigma|$.

$\ell \rightarrow$ Denotes the word length in a block code $\mathcal{X}$.

$\Sigma^\ell \rightarrow$ Denotes the set of all words of length ℓ with letters in Σ.

$\Sigma^* \rightarrow$ Denotes the set of all finite words with letters in Σ.

$$\Sigma^* := \bigcup_{\ell \geq 0} \Sigma^\ell$$

$\Sigma^+ \rightarrow$ Denotes the set of all not empty finite words with letters in Σ.

$$\Sigma^+ := \bigcup_{\ell \in \mathbb{N}} \Sigma^\ell$$

$w \rightarrow$ Denotes a tuple of letters. Assume $w = (b_1 b_2 \dots b_\ell)$, where $b_i \in \Sigma$, it follows that $w \in \Sigma^\ell$.

$\mathcal{X} \rightarrow$ Denotes a code. A code is a set of words so that any concatenation $w_1 \dots w_n$ of words in $w_1, \dots, w_n \in \mathcal{X}$ has a unique decomposition into words from $\mathcal{X}$

$M(n, \ell) \rightarrow$ Denotes the function that returns the maximum size of a circular code with respect to $n = |\Sigma|$ and the word length ℓ

$\mathcal{X}^x \rightarrow$ Denotes the x-ary Cartesian power of a code $\mathcal{X}$ i.e. $\mathcal{X}^2$ is the cartesian product of $\mathcal{X} \times \mathcal{X}$

$\overleftarrow{\cdot} \rightarrow$ Denotes the reversing permutation, *i.e.* for a word $w \coloneqq b_1 \cdots b_\ell$ we have:

$$\overleftarrow{b_1 \dots b_\ell} \coloneqq b_\ell \dots b_1$$

Let x be a positive integer and $\mathcal{X} = \{w_1, w_2, \dots, w_x\} \subset \Sigma^*$ be code over an arbitrary finite alphabet Σ, then $\overleftarrow{\mathcal{X}} \coloneqq \{\overleftarrow{w_1}, \overleftarrow{w_2}, \dots, \overleftarrow{w_x}\}$

$\alpha_j(\cdot) \rightarrow$ Denotes the circular permutation by j letters of a word. Let $w = b_1 \dots b_\ell \in \Sigma^\ell$ be a word of length ℓ over an arbitrary alphabet Σ and $j \geq 1$ integer. Then:

$$\alpha_j(w) \coloneqq b_{j+1} \dots b_\ell b_j \dots b_j$$

Let x be a positive integer and $\mathcal{X} = \{w_1, w_2, \dots, w_x\} \subset \Sigma^*$ be code over an arbitrary finite alphabet Σ, then $\alpha_j(\mathcal{X}) \coloneqq \{\alpha_j(w_1), \alpha_j(w_2), \dots, \alpha_j(w_x)\}$

Nitrogenous bases

$\mathfrak{B} \rightarrow$ Denotes the genetic alphabet $\mathfrak{B} \coloneqq \{A, T, G, C\}$.

$\mathfrak{B}^2 \rightarrow$ Denotes the set of all 16 dinucleotides.

$\mathfrak{B}^3 \rightarrow$ Denotes the set of all 64 trinucleotides.

$\mathfrak{B}^4 \rightarrow$ Denotes the set of all 256 tetranucleotides.

$c(\cdot) \rightarrow$ Denotes the complementary mapping of nitrogenous bases.

$$c(T) \coloneqq A \textit{ and } c(C) \coloneqq G \textit{ and vice versa}$$

Let x be a positive integer and $\mathcal{X} = \{w_1, w_2, \dots, w_x\} \subset \mathfrak{B}^*$ be a code, then $c(\mathcal{X}) \coloneqq \{c(w_1), c(w_2), \dots, c(w_x)\}$

Logical operators

$\wedge \rightarrow$ Denotes the logical *and*; $a \wedge b$ means that both, a and b, have to be satisfied.

$\vee \rightarrow$ Denotes the logical *or*; $a \vee b$ means that both or at least one of both, a or b, has to be satisfied.

$\dot{\vee} \rightarrow$ Denotes the logical *exclusive or*; $a \dot{\vee} b$ means that either a or b has to be satisfied but not both.

Chapter II

Appendix

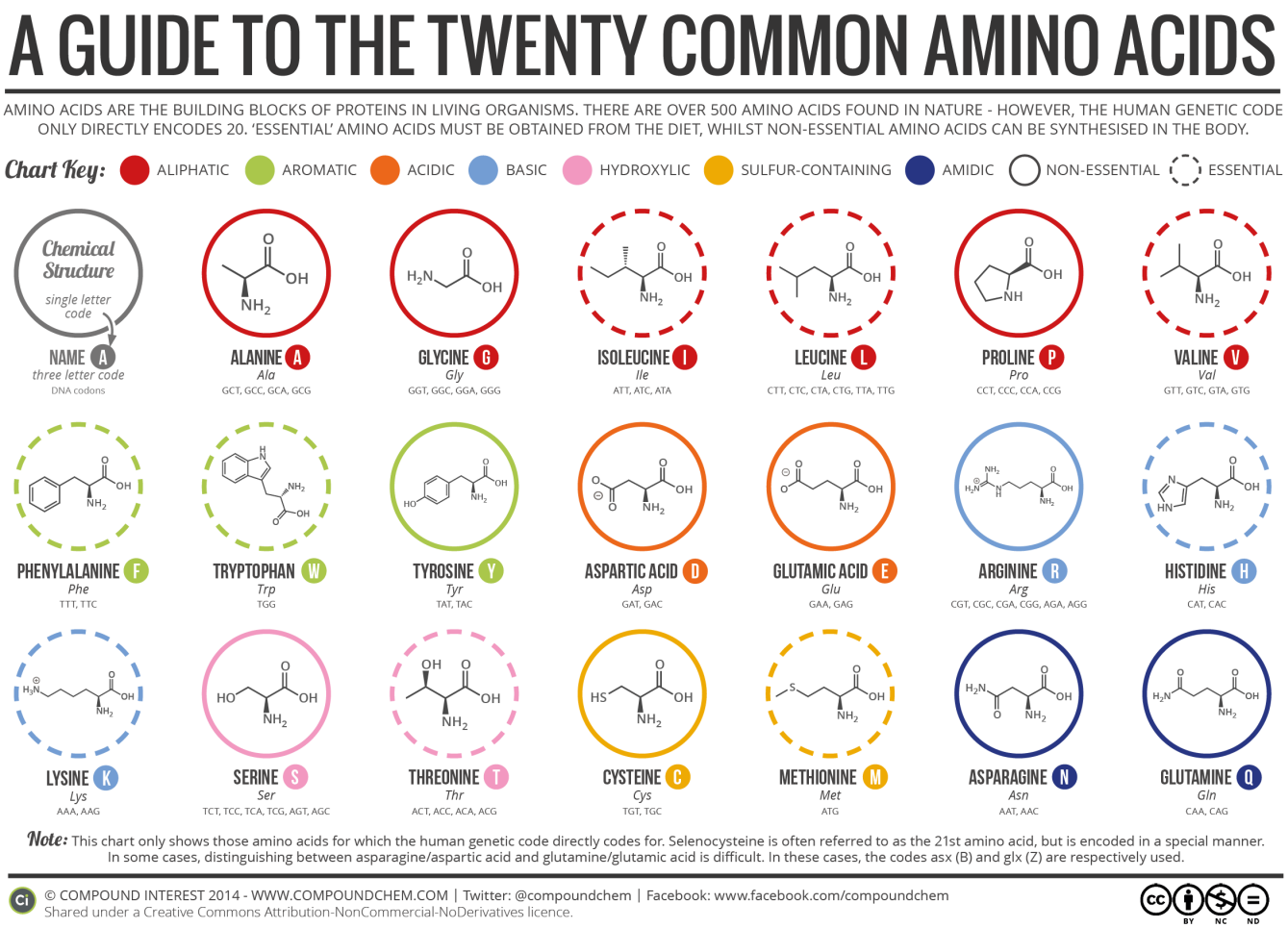


Figure II.1: A list of all 20 amino acids encoded in the standard genetic code. Including the structure, name and acronym. (Picture taken from www.compoundchem.com, shared under a Creative Commons license)

II.1 Proof of Theorem 2.1.4

Proof. Let $\mathcal{X} \subseteq \mathfrak{B}^3$ be a maximal self-complementary circular code and $\mathcal{G}(\mathcal{X})$ its associated graph. Since $\mathcal{X}$ is circular then $\mathcal{G}(\mathcal{X})$ is acyclic, so it has a path $p = p_{max}(\mathcal{X})$ of maximal length $l(p)$.

Claim (1): Assume that $p = d_1 \to b_1 \to \cdots \to b_k$, then any concatenation $d_i b_i \in \mathcal{X}$. Choose any trinucleotide $c = s_1 s_2 s_3 \in \mathcal{X}$. Then

$$(d_1 b_1) \dots (d_k b_k)(s_1 s_2 s_3) \in \mathcal{X}^{k+1}$$

and hence $(d_1 b_1) \dots (d_k b_k) s_1$ is a possible $\mathcal{X}$-frame (for itself) with $t_b = \epsilon$ and $t_e = s_1$. Moreover, each concatenation $b_i d_{i+1}$ is also a trinucleotide in $\mathcal{X}$, so $d_1 (b_1 d_2) \dots (b_{k-1} d_k) b_k s_1$ is a second possible $\mathcal{X}$-frame with $t_b = d_1$ and $t_e = b_k s_1$. Thus $n_{\mathcal{X}} \geq l_w(p) + 2$ since the sequence $d_1 b_1 \dots d_k b_k s_1$ has length $l_w(p) + 1$.

Now assume that $b_1 \dots b_k$ is a sequence of nucleotides and assume that $k \geq l_w(p) + 2$ but $b_1 \dots b_k$ has 2 different possible $\mathcal{X}$-frames. We have to show a contradiction to conclude that $n_{\mathcal{X}} = l_w(p) + 2$. Assume that $t_b u_1 \dots u_l t_e$ and $t'_b u'_1 \dots u'_m t'_e$ with $u_i, u'_i \in \mathcal{X}$ and $t_b, t_e, t'_b, t'_e \in (\{\epsilon\} \cup \mathfrak{B} \cup \mathfrak{B}^2)$ are the 2 different possible $\mathcal{X}$-frames. Obviously, $\mid t_b t_e \mid \leq 4$. If $\mid t_b t_e \mid = 4$ then by the difference of the 2 possible $\mathcal{X}$-frames, we conclude that at least one of t'_b or t'_e has to have length ≥ 3, a contradiction to the definition of possible $\mathcal{X}$-frame, or $\mid t'_b t'_e \mid \leq 3$. Hence w.l.o.g. we assume that $\mid t_b t_e \mid \leq 3$. Consequently, $\mid u_1 \dots u_l \mid \geq k - 3 \geq l_w(p) + 2 - 3 = l_w(p) - 1$ and hence $\mid u_1 \dots u_l \mid \geq l_w(p) + 1$. We now have to distinguish cases:

(a) If $\mid t_b t_e \mid \leq 1$ then we even get $\mid u_1 \dots u_l \mid \geq k - 1 \geq l_w(p) + 2 - 1 = l_w(p) + 1$ and hence $\mid u_1 \dots u_l \mid \geq l_w(p)$. Thus the path associated to the 2 possible $\mathcal{X}$-frames has word-length at least $l_w(p)+1$, a contradiction to the maximality of $l_w(p)$. In this case, the sequence $u_1 \dots u_l$ could contain the sequence $u'_1 \dots u'_m$ as a subsequence.

(b) If $\mid t_b t_e \mid \geq 2$ then the second possible $\mathcal{X}$-frame is at least shifted by one with respect to the first possible $\mathcal{X}$-frame, i.e. it must extend the sequence $u_1 \dots u_l$ to the left or to the right. In this case, the sequence $u_1 \dots u_l$ cannot contain the sequence $u'_1 \dots u'_m$ as a subsequence. The path associated to the 2 possible $\mathcal{X}$-frames has word-length at least $\mid u_1 \dots u_l \mid +1 \geq l_w(p) + 1$, again a contradiction to the maximality of $l_w(p)$.

Thus, $n_{\mathcal{X}} = l(p) + 2$.

The case $p = b_1 \to d_1 \to \cdots \to d_k$ is symmetric and can be similarly dealt with.

Claim (2): Assume that $p = d_1 \to b_1 \to \cdots \to d_k$, then any concatenation $d_i b_i \in \mathcal{X}$. As in Claim (1), $(d_1 b_1) \dots (d_{k-1} b_{k-1}) d_k$ is a possible $\mathcal{X}$-frame (for itself)

with $t_b = \epsilon$ and $t_e = d_k$. Moreover, each concatenation $b_i d_{i+1}$ is a trinucleotide in $\mathcal{X}$, so $d_1(b_1d_2)\dots(b_{k-2}d_{k-1})(b_{k-1}d_k)$ is a second possible $\mathcal{X}$-frame with $t_b = d_1$ and $t_e = \epsilon$. Thus $n_{\mathcal{X}} \geq l_w(p)$ since the sequence $d_1b_1 \dots d_{k-1}b_{k-1}d_k$ has length $l_w(p)$.

Now assume that $b_1 \dots b_k$ is a sequence of nucleotides and assume that $k \geq l_w(p)+1$ but $b_1 \dots b_k$ has 2 different possible $\mathcal{X}$-frames: $t_b u_1 \dots u_l t_e$ and $t'_b u'_1 \dots u'_m t'_e$ with $u_i, u'_i \in \mathcal{X}$ and $t_b, t_e, t'_b, t'_e \in (\{\epsilon\} \cup \mathfrak{B} \cup \mathfrak{B}^2)$. As in Claim (1), we assume w.l.o.g. that $\mid t_b t_e \mid \leq 3$. We distinguish cases:

(a) If $\mid t_b t_e \mid = 0$ then $\mid u_1 \dots u_l \mid \geq l_w(p) + 1$ and $u'_1 \dots u'_m$ is a subsequence of $u_1 \dots u_l$. Thus the path associated to the 2 possible $\mathcal{X}$-frames has word-length $l_w(p) + 1$ with the associated word $u_1 \dots u_l$, a contradiction to the maximality of $l_w(p)$.

(b) If $\mid t_b t_e \mid = 1$ then $\mid u_1 \dots u_l \mid \geq l_w(p)$. If the second possible $\mathcal{X}$-frame is shifted by one with respect to the first one, then the path associated to the 2 possible $\mathcal{X}$-frames has word-length $l_w(p) + 1$, again a contradiction to the maximality of $l_w(p)$. If the second possible $\mathcal{X}$-frame is shifted by two, then the path associated to the 2 possible $\mathcal{X}$-frames has $u_1 \dots u_l$ has word-length $l_w(p)$. However, in this case, the path starts with a dinucleotide and ends with a nucleotide, a contradiction to the structure of maximal paths which have to start and end with a dinucleotide.

(c) If $\mid t_b t_e \mid = 2$ then $\mid u_1 \dots u_l \mid \geq l_w(p) - 1$. Again, we have to distinguish cases:

  (i) $\mid t_b \mid = 2$ and $\mid t_e \mid = 0$. Then the associated path to the 2 possible $\mathcal{X}$-frames has word-length $l_w(p)$ and starts with a nucleotide but ends with a dinucleotide, a contradiction to the structure of maximal paths, or has word-length $l_w(p)+1$, a contradiction to the maximality of $l_w(p)$.

  (ii) $\mid t_b \mid = 0$ and $\mid t_e \mid = 2$, as (i).

  (iii) $\mid t_b \mid = 1$ and $\mid t_e \mid = 1$. As above, if the second possible $\mathcal{X}$-frame is shifted by one, then the path associated to the 2 possible $\mathcal{X}$-frames has word-length $l_w(p)$ again starting with a nucleotide (u_1) and ending with a dinucleotide, a contradiction to the structure of maximal paths. If the second possible $\mathcal{X}$-frame is shifted by two, then again the path associated to the 2 possible $\mathcal{X}$-frames has word-length $l_w(p)$ starting with a nucleotide (u'_1) and ends with a dinucleotide.

(d) If $\mid t_b t_e \mid = 3$ then $\mid u_1 \dots u_l \mid \geq l_w(p) - 2$. We distinguish two symmetric cases:

  (i) $\mid t_b \mid = 2$ and $\mid t_e \mid = 1$. If the second possible $\mathcal{X}$-frame is shifted by one, then the path associated to the 2 possible $\mathcal{X}$-frames has word-length

$l_w(p)+1$, a contradiction to the maximality of $l_w(p)$, or has word-length $l_w(p)$ but starting with a nucleotide and ending with a dinucleotide, a contradiction to the structure of maximal paths. If the second possible $\mathcal{X}$-frame is shifted by two, then either the path associated to the 2 possible $\mathcal{X}$-frames has word-length $l_w(p)+1$, a contradiction to the maximality of $l_w(p)$, or has word-length $l_w(p)-1$ starting with a nucleotide and ending with a nucleotide. But this case cannot exist unless the arrow-length of this path is at least the arrow-length of p, a contradiction to the maximality of p.

(ii) $\mid t_b \mid = 1$ and $\mid t_e \mid = 2$, as (i).

Claim (3): Assume that $p = b_1 \to d_1 \to \cdots \to b_k$, then any concatenation $b_i d_i \in \mathcal{X}$. Choose any 2 trinucleotides $c = s_1 s_2 s_3, c' = s'_1 s'_2 s'_3 \in \mathcal{X}$. Then

$$(s'_1 s'_2 s'_3)(b_1 d_1)\dots(d_k b_k)(s_1 s_2 s_3) \in \mathcal{X}^{k+2}$$

and hence $s'_3(b_1 d_1)\dots(b_{k-1}d_{k-1})b_k s_1$ is a possible $\mathcal{X}$-frame (for itself) with $t_b = s'_3$ and $t_e = b_k s_1$. Moreover, each concatenation $d_i b_{i+1}$ is a trinucleotide in $\mathcal{X}$, so $s'_3 b_1(d_1 b_2)\dots(d_{k-1}b_k)s_1$ is a second possible $\mathcal{X}$-frame with $t_b = s'_3 b_1$ and $t_e = s_1$. Thus $n_{\mathcal{X}} \geq l_w(p)+3$ since the sequence $s'_3 b_1 d_1 \dots b_{k-1}d_{k-1}b_k s_1$ has length $l_w(p)+2$.

Now assume that $b_1 \dots b_k$ is a sequence of nucleotides with $k \geq l_w(p)+3$ but $b_1 \dots b_k$ has 2 different possible $\mathcal{X}$-frames: $t_b u_1 \dots u_l t_e$ and $t'_b u'_1 \dots u'_m t'_e$ with $u_i, u'_i \in \mathcal{X}$ and $t_b, t_e, t'_b, t'_e \in (\{\epsilon\} \cup \mathfrak{B} \cup \mathfrak{B}^2)$. As in Claim (1), we conclude that w.l.o.g. $\mid t_b t_e \mid \leq 3$ and hence $\mid u_1 \dots u_l \mid \geq k-3 \geq l_w(p)+3-3 = l_w(p)$. Similar arguments as above show that the path associated to the 2 possible $\mathcal{X}$-frames has word-length greater than $l_w(p)$, in contradiction to the maximality of p and $l_w(p)$. □

II.2 List of all representing tables for all maximal circular Tessera codes

List II.2.1.

For fragment distribution: $|\mathcal{X}_1| = 0$, $|\mathcal{X}_2| = 2$ $|\mathcal{X}_3| = 4$ *and* $|\mathcal{X}_4| = 6$ *(1)*

0	*0*	*0*	*0*
2	*2*	*0*	*0*
2	*2*	*4*	*0*
2	*2*	*2*	*6*

Number 1) $P_o = \frac{1}{4!}$*; 32256 Codes*

For fragment distribution: $|\mathcal{X}_1| = 1$, $|\mathcal{X}_2| = 1$ $|\mathcal{X}_3| = 4$ *and* $|\mathcal{X}_4| = 6$ *(1)*

1	*1*	*0*	*0*
1	*1*	*0*	*0*
2	*2*	*4*	*0*
2	*2*	*2*	*6*

Number 2) $P_o = \frac{2}{4!}$*; 32256 Codes*

For fragment distribution: $|\mathcal{X}_1| = 0$, $|\mathcal{X}_2| = 3$ $|\mathcal{X}_3| = 3$ *and* $|\mathcal{X}_4| = 6$ *(1)*

0	*0*	*0*	*0*
2	*3*	*1*	*0*
2	*1*	*3*	*0*
2	*2*	*2*	*6*

Number 3) $P_o = \frac{2}{4!}$*; 36864 Codes*

For fragment distribution: $|\mathcal{X}_1| = 1$, $|\mathcal{X}_2| = 2$ $|\mathcal{X}_3| = 3$ *and* $|\mathcal{X}_4| = 6$ *(2)*

1	*1*	*0*	*0*
1	*3*	*2*	*0*
2	*0*	*2*	*0*
2	*2*	*2*	*6*

Number 4) $P_o = \frac{1}{4!}$*; 73728 Codes*

1	*1*	*0*	*0*
1	*2*	*1*	*0*
2	*1*	*3*	*0*
2	*2*	*2*	*6*

Number 5) $P_o = \frac{1}{4!}$*; 147456 Codes*

For fragment distribution: $|\mathcal{X}_1| = 2$, $|\mathcal{X}_2| = 2$ $|\mathcal{X}_3| = 2$ *and* $|\mathcal{X}_4| = 6$ *(2)*

2	*2*	*0*	*0*
0	*2*	*2*	*0*
2	*0*	*2*	*0*
2	*2*	*2*	*6*

Number 6) $P_o = \frac{3}{4!}$*; 12288 Codes*

2	*1*	*1*	*0*
1	*2*	*1*	*0*
1	*1*	*2*	*0*
2	*2*	*2*	*6*

Number 7) $P_o = \frac{6}{4!}$*; 49152 Codes*

For fragment distribution: $|\mathcal{X}_1| = 0$, $|\mathcal{X}_2| = 2$ $|\mathcal{X}_3| = 5$ *and* $|\mathcal{X}_4| = 5$ *(1)*

0	*0*	*0*	*0*
2	*2*	*0*	*0*
2	*2*	*5*	*1*
2	*2*	*1*	*5*

Number 8) $P_o = \frac{2}{4!}$*; 31104 Codes*

For fragment distribution: $|\mathcal{X}_1| = 1$, $|\mathcal{X}_2| = 1$ $|\mathcal{X}_3| = 5$ *and* $|\mathcal{X}_4| = 5$ *(1)*

1	*1*	*0*	*0*
1	*1*	*0*	*0*
2	*2*	*5*	*1*
2	*2*	*1*	*5*

Number 9) $P_o = \frac{4}{4!}$*; 31104 Codes*

For fragment distribution: $|\mathcal{X}_1| = 0$, $|\mathcal{X}_2| = 3$ $|\mathcal{X}_3| = 4$ *and* $|\mathcal{X}_4| = 5$ *(2)*

0	0	0	0
2	3	1	0
2	1	5	2
2	2	0	4

Number 10) $P_o = \frac{1}{4!}$*; 96768 Codes*

0	0	0	0
2	3	1	0
2	1	4	1
2	2	1	5

Number 11) $P_o = \frac{1}{4!}$*; 165888 Codes*

For fragment distribution: $|\mathcal{X}_1| = 1$, $|\mathcal{X}_2| = 2$ $|\mathcal{X}_3| = 4$ *and* $|\mathcal{X}_4| = 5$ *(4)*

1	1	0	0
1	5	2	2
2	0	2	0
2	0	2	4

Number 12) $P_o = \frac{1}{4!}$*; 96768 Codes*

1	1	0	0
1	4	2	1
2	0	2	0
2	1	2	5

Number 13) $P_o = \frac{1}{4!}$*; 165888 Codes*

1	1	0	0
1	2	1	0
2	1	5	2
2	2	0	4

Number 14) $P_o = \frac{1}{4!}$*; 193536 Codes*

1	1	0	0
1	2	1	0
2	1	4	1
2	2	1	5

Number 15) $P_o = \frac{1}{4!}$*; 331776 Codes*

For fragment distribution: $|\mathcal{X}_1| = 1$, $|\mathcal{X}_2| = 3$ $|\mathcal{X}_3| = 3$ *and* $|\mathcal{X}_4| = 5$ *(3)*

1	1	0	0
1	5	2	2
2	0	3	1
2	0	1	3

Number 16) $P_o = \frac{2}{4!}$*; 110592 Codes*

1	1	0	0
1	3	2	0
2	0	3	1
2	2	1	5

Number 17) $P_o = \frac{1}{4!}$*; 221184 Codes*

1	*1*	*0*	*0*
1	*3*	*1*	*1*
2	*1*	*3*	*0*
2	*1*	*2*	*5*

Number 18) $P_o = \frac{1}{4!}$*; 387072 Codes*

For fragment distribution: $|\mathcal{X}_1| = 2$, $|\mathcal{X}_2| = 2$ $|\mathcal{X}_3| = 3$ *and* $|\mathcal{X}_4| = 5$ *(4)*

2	*1*	*1*	*0*
1	*3*	*2*	*0*
1	*0*	*2*	*1*
2	*2*	*1*	*5*

Number 19) $P_o = \frac{1}{4!}$*; 442368 Codes*

2	*2*	*0*	*0*
0	*2*	*2*	*0*
2	*0*	*3*	*1*
2	*2*	*1*	*5*

Number 20) $P_o = \frac{1}{4!}$*; 110592 Codes*

2	*2*	*0*	*0*
0	*2*	*1*	*1*
2	*1*	*3*	*0*
2	*1*	*2*	*5*

Number 21) $P_o = \frac{1}{4!}$*; 221184 Codes*

2	*1*	*1*	*0*
1	*2*	*1*	*0*
1	*1*	*3*	*1*
2	*2*	*1*	*5*

Number 22) $P_o = \frac{2}{4!}$*; 387072 Codes*

For fragment distribution: $|\mathcal{X}_1| = 0$, $|\mathcal{X}_2| = 4$ $|\mathcal{X}_3| = 4$ *and* $|\mathcal{X}_4| = 4$ *(2)*

0	*0*	*0*	*0*
2	*4*	*2*	*0*
2	*0*	*4*	*2*
2	*2*	*0*	*4*

Number 23) $P_o = \frac{3}{4!}$*; 21952 Codes*

0	*0*	*0*	*0*
2	*4*	*1*	*1*
2	*1*	*4*	*1*
2	*1*	*1*	*4*

Number 24) $P_o = \frac{6}{4!}$*; 55296 Codes*

For fragment distribution: $|\mathcal{X}_1| = 1$, $|\mathcal{X}_2| = 3$ $|\mathcal{X}_3| = 4$ *and* $|\mathcal{X}_4| = 4$ *(4)*

1	*1*	*0*	*0*
1	*4*	*2*	*1*
2	*0*	*4*	*2*
2	*1*	*0*	*3*

Number 25) $P_o = \frac{1}{4!}$*; 258048 Codes*

1	1	0	0
1	4	2	1
2	0	3	1
2	1	1	4

Number 26) $P_o = \frac{1}{4!}$*; 442368 Codes*

1	1	0	0
1	3	2	0
2	0	4	2
2	2	0	4

Number 27) $P_o = \frac{1}{4!}$*; 150528 Codes*

1	1	0	0
1	3	1	1
2	1	4	1
2	1	1	4

Number 28) $P_o = \frac{2}{4!}$*; 387072 Codes*

For fragment distribution: $|\mathcal{X}_1| = 2$, $|\mathcal{X}_2| = 2$ $|\mathcal{X}_3| = 4$ *and* $|\mathcal{X}_4| = 4$ *(4)*

2	2	0	0
0	2	2	0
2	0	4	2
2	2	0	4

Number 29) $P_o = \frac{1}{4!}$*; 75264 Codes*

2	2	0	0
0	2	1	1
2	1	4	1
2	1	1	4

Number 30) $P_o = \frac{2}{4!}$*; 221184 Codes*

2	1	1	0
1	2	1	0
1	1	4	2
2	2	0	4

Number 31) $P_o = \frac{2}{4!}$*; 258048 Codes*

2	1	1	0
1	2	0	1
1	2	4	1
2	1	1	4

Number 32) $P_o = \frac{2}{4!}$*; 442368 Codes*

For fragment distribution: $|\mathcal{X}_1| = 2$, $|\mathcal{X}_2| = 3$ $|\mathcal{X}_3| = 3$ *and* $|\mathcal{X}_4| = 4$ *(7)*

2	2	0	0
0	4	2	2
2	0	3	1
2	0	1	3

Number 33) $P_o = \frac{2}{4!}$*; 86016 Codes*

2	1	1	0
1	4	2	1
1	0	3	2
2	1	0	3

Number 34) $P_o = \frac{1}{4!}$*; 589824 Codes*

2	2	0	0
0	3	2	1
2	0	4	2
2	1	0	3

Number 35) $P_o = \frac{1}{4!}$*; 172032 Codes*

2	1	1	0
1	3	1	1
1	1	4	2
2	1	0	3

Number 36) $P_o = \frac{1}{4!}$*; 1032192 Codes*

2	2	0	0
0	3	2	1
2	0	3	1
2	1	1	4

Number 37) $P_o = \frac{1}{4!}$*; 294912 Codes*

2	1	1	0
1	3	2	0
1	0	3	2
2	2	0	4

Number 38) $P_o = \frac{1}{4!}$*; 344064 Codes*

2	1	1	0
1	3	1	1
1	1	3	1
2	1	1	4

Number 39) $P_o = \frac{2}{4!}$*; 903168 Codes*

For fragment distribution: $|\mathcal{X}_1| = 3$, $|\mathcal{X}_2| = 3$ $|\mathcal{X}_3| = 3$ *and* $|\mathcal{X}_4| = 3$ *(3)*

3	2	1	0
0	3	2	1
1	0	3	2
2	1	0	3

Number 40) $P_o = \frac{4}{4!}$*; 98304 Codes*

3	2	1	0
0	3	1	2
1	1	3	1
2	0	1	3

Number 41) $P_o = \frac{3}{4!}$*; 229376 Codes*

3	1	1	1
1	3	1	1
1	1	3	1
1	1	1	3

Number 42) $P_o = 1$*; 153600 Codes*

In total 9592512 Codes

II.3 List of 52 maximal 1-circular codes encoding all 20 amino acids.

The list below gives all 52 perfect matchings of the graph $\mathcal{G}$ from Lemma 4.4.1. The amino acids are listed in the order of their assignments to the 13 complete equivalence classes $D_1, \ldots, D_{20}$.

Code: 01 $D_1 \ldots D_{20}$: Asp Ile His Thr Glu Val Ser Tyr Ala Pro Arg Cys Leu
Code: 02 $D_1 \ldots D_{20}$: Asp Ile His Thr Glu Val Ser Tyr Ala Pro Arg Leu Cys
Code: 03 $D_1 \ldots D_{20}$: Asp Ile His Thr Glu Val Ser Tyr Pro Leu Arg Ala Cys
Code: 04 $D_1 \ldots D_{20}$: Asp Ile Pro Thr Glu Ser His Tyr Ala Leu Arg Cys Val
Code: 05 $D_1 \ldots D_{20}$: Asp Ile Pro Thr Glu Ser His Tyr Arg Leu Val Ala Cys
Code: 06 $D_1 \ldots D_{20}$: Asp Ile Pro Thr Glu Val His Tyr Ala Ser Arg Cys Leu
Code: 07 $D_1 \ldots D_{20}$: Asp Ile Pro Thr Glu Val His Tyr Ala Ser Arg Leu Cys
Code: 08 $D_1 \ldots D_{20}$: Asp Ile Pro Thr Glu Val His Tyr Arg Leu Ser Ala Cys
Code: 09 $D_1 \ldots D_{20}$: Asp Ile Thr Leu Glu Ser His Tyr Ala Pro Arg Cys Val
Code: 10 $D_1 \ldots D_{20}$: Asp Ile Thr Leu Glu Ser His Tyr Arg Pro Val Ala Cys
Code: 11 $D_1 \ldots D_{20}$: Asp Ile Thr Leu Glu Val His Tyr Arg Pro Ser Ala Cys
Code: 12 $D_1 \ldots D_{20}$: Asp Ile Thr Leu Glu Val His Tyr Pro Ser Arg Ala Cys
Code: 13 $D_1 \ldots D_{20}$: Asp Ile Thr Tyr Glu Ser His Leu Ala Pro Arg Cys Val
Code: 14 $D_1 \ldots D_{20}$: Asp Ile Thr Tyr Glu Ser His Leu Arg Pro Val Ala Cys
Code: 15 $D_1 \ldots D_{20}$: Asp Ile Thr Tyr Glu Val His Leu Arg Pro Ser Ala Cys
Code: 16 $D_1 \ldots D_{20}$: Asp Ile Thr Tyr Glu Val His Leu Pro Ser Arg Ala Cys
Code: 17 $D_1 \ldots D_{20}$: Glu Asp His Thr Ala Ser Ile Tyr Arg Pro Val Cys Leu
Code: 18 $D_1 \ldots D_{20}$: Glu Asp His Thr Ala Ser Ile Tyr Arg Pro Val Leu Cys
Code: 19 $D_1 \ldots D_{20}$: Glu Asp His Thr Ala Ser Ile Tyr Pro Leu Arg Cys Val
Code: 20 $D_1 \ldots D_{20}$: Glu Asp His Thr Ala Val Ile Tyr Arg Pro Ser Cys Leu
Code: 21 $D_1 \ldots D_{20}$: Glu Asp His Thr Ala Val Ile Tyr Arg Pro Ser Leu Cys
Code: 22 $D_1 \ldots D_{20}$: Glu Asp His Thr Ala Val Ile Tyr Pro Ser Arg Cys Leu
Code: 23 $D_1 \ldots D_{20}$: Glu Asp His Thr Ala Val Ile Tyr Pro Ser Arg Leu Cys
Code: 24 $D_1 \ldots D_{20}$: Glu Asp His Thr Ser Val Ile Tyr Ala Pro Arg Cys Leu
Code: 25 $D_1 \ldots D_{20}$: Glu Asp His Thr Ser Val Ile Tyr Ala Pro Arg Leu Cys
Code: 26 $D_1 \ldots D_{20}$: Glu Asp His Thr Ser Val Ile Tyr Pro Leu Arg Ala Cys
Code: 27 $D_1 \ldots D_{20}$: Glu Asp Thr Tyr Ala Ser His Ile Arg Pro Val Cys Leu
Code: 28 $D_1 \ldots D_{20}$: Glu Asp Thr Tyr Ala Ser His Ile Arg Pro Val Leu Cys
Code: 29 $D_1 \ldots D_{20}$: Glu Asp Thr Tyr Ala Ser His Ile Pro Leu Arg Cys Val
Code: 30 $D_1 \ldots D_{20}$: Glu Asp Thr Tyr Ala Val His Ile Arg Pro Ser Cys Leu
Code: 31 $D_1 \ldots D_{20}$: Glu Asp Thr Tyr Ala Val His Ile Arg Pro Ser Leu Cys
Code: 32 $D_1 \ldots D_{20}$: Glu Asp Thr Tyr Ala Val His Ile Pro Ser Arg Cys Leu
Code: 33 $D_1 \ldots D_{20}$: Glu Asp Thr Tyr Ala Val His Ile Pro Ser Arg Leu Cys
Code: 34 $D_1 \ldots D_{20}$: Glu Asp Thr Tyr Ser Val His Ile Ala Pro Arg Cys Leu

Code: 35 $D_1 \dots D_{20}$: Glu Asp Thr Tyr Ser Val His Ile Ala Pro Arg Leu Cys
Code: 36 $D_1 \dots D_{20}$: Glu Asp Thr Tyr Ser Val His Ile Pro Leu Arg Ala Cys
Code: 37 $D_1 \dots D_{20}$: Thr Asp His Leu Glu Ser Ile Tyr Ala Pro Arg Cys Val
Code: 38 $D_1 \dots D_{20}$: Thr Asp His Leu Glu Ser Ile Tyr Arg Pro Val Ala Cys
Code: 39 $D_1 \dots D_{20}$: Thr Asp His Leu Glu Val Ile Tyr Arg Pro Ser Ala Cys
Code: 40 $D_1 \dots D_{20}$: Thr Asp His Leu Glu Val Ile Tyr Pro Ser Arg Ala Cys
Code: 41 $D_1 \dots D_{20}$: Thr Asp His Tyr Glu Ser Ile Leu Ala Pro Arg Cys Val
Code: 42 $D_1 \dots D_{20}$: Thr Asp His Tyr Glu Ser Ile Leu Arg Pro Val Ala Cys
Code: 43 $D_1 \dots D_{20}$: Thr Asp His Tyr Glu Val Ile Leu Arg Pro Ser Ala Cys
Code: 44 $D_1 \dots D_{20}$: Thr Asp His Tyr Glu Val Ile Leu Pro Ser Arg Ala Cys
Code: 45 $D_1 \dots D_{20}$: Thr Asp His Tyr Glu Val Ser Ile Ala Pro Arg Cys Leu
Code: 46 $D_1 \dots D_{20}$: Thr Asp His Tyr Glu Val Ser Ile Ala Pro Arg Leu Cys
Code: 47 $D_1 \dots D_{20}$: Thr Asp His Tyr Glu Val Ser Ile Pro Leu Arg Ala Cys
Code: 48 $D_1 \dots D_{20}$: Thr Asp Pro Tyr Glu Ser His Ile Ala Leu Arg Cys Val
Code: 49 $D_1 \dots D_{20}$: Thr Asp Pro Tyr Glu Ser His Ile Arg Leu Val Ala Cys
Code: 50 $D_1 \dots D_{20}$: Thr Asp Pro Tyr Glu Val His Ile Ala Ser Arg Cys Leu
Code: 51 $D_1 \dots D_{20}$: Thr Asp Pro Tyr Glu Val His Ile Ala Ser Arg Leu Cys
Code: 52 $D_1 \dots D_{20}$: Thr Asp Pro Tyr Glu Val His Ile Arg Leu Ser Ala Cys

The list below gives all 52 maximal 1-circular codes encoding all 20 amino acids in lexicographical order.

$\{AAC, AAG, ACC, AGT, ATA, ATG, CAG, CAT, CCT, CGC, CTA, GAC, GAG, GCT, GGC, GTC, TAT, TGG, TGT, TTC\}$
$\{AAC, AAG, ACC, AGT, ATA, ATG, CAG, CAT, CCT, CGC, GAC, GAG, GCT, GGC, GTC, TAC, TGG, TGT, TTA, TTC\}$
$\{AAC, AAG, ACC, AGT, ATA, ATG, CAG, CAT, CCT, CGT, CTA, GAC, GAG, GCC, GGC, GTT, TAT, TGC, TGG, TTC\}$
$\{AAC, AAG, ACC, AGT, ATA, ATG, CAG, CAT, CCT, CGT, GAC, GAG, GCC, GGC, GTT, TAC, TGC, TGG, TTA, TTC\}$
$\{AAC, AAG, ACC, ATA, ATG, CAG, CAT, CCG, CGT, CTA, GAC, GAG, GCT, GGC, GTA, TAT, TCC, TGG, TGT, TTC\}$
$\{AAC, AAG, ACC, ATA, ATG, CAG, CAT, CCG, CGT, GAC, GAG, GCT, GGC, GTA, TAC, TCC, TGG, TGT, TTA, TTC\}$
$\{AAC, AAG, ACC, ATA, ATG, CAG, CAT, CCT, CGC, CTA, GAC, GAG, GCT, GGC, GTA, TAT, TCG, TGG, TGT, TTC\}$
$\{AAC, AAG, ACC, ATA, ATG, CAG, CAT, CCT, CGC, GAC, GAG, GCT, GGC, GTA, TAC, TCG, TGG, TGT, TTA, TTC\}$
$\{AAC, AAG, ACT, AGT, ATA, ATG, CAG, CAT, CCA, CGC, CTC, GAC, GAG, GCT, GGC, GTC, TAT, TGG, TGT, TTC\}$
$\{AAC, AAG, ACT, AGT, ATA, ATG, CAG, CAT, CCA, CGT, CTC, GAC, GAG, GCC, GGC, GTT, TAT, TGC, TGG, TTC\}$
$\{AAC, AAG, ACT, ATA, ATG, CAC, CAG, CCG, CGT, CTC, GAC, GAG, GCT, GGC, GTA, TAT, TCA, TGG, TGT, TTC\}$
$\{AAC, AAG, ACT, ATA, ATG, CAC, CAG, CCT, CGT, CTG, GAC, GAG, GCC, GGC, GTA, TAT, TCA, TGG, TGT, TTC\}$
$\{AAC, AAG, ACT, ATA, ATG, CAC, CAG, CCT, CGT, GAC, GAG, GCC, GGC, GTA, TAT, TCA, TGC, TGG, TTC, TTG\}$
$\{AAC, AAG, ACT, ATA, ATG, CAG, CAT, CCA, CGC, CTC, GAC, GAG, GCT, GGC, GTA, TAT, TCG, TGG, TGT, TTC\}$
$\{AAC, AAG, ACT, ATA, ATG, CAG, CAT, CCA, CGT, CTG, GAC, GAG, GCC, GGC, GTA, TAT, TCC, TGG, TGT, TTC\}$
$\{AAC, AAG, ACT, ATA, ATG, CAG, CAT, CCA, CGT, GAC, GAG, GCC, GGC, GTA, TAT, TCC, TGC, TGG, TTC, TTG\}$
$\{AAG, AAT, ACA, AGT, ATC, ATG, CAC, CAG, CCT, CGC, CTA, GAC, GAG, GCT, GGC, GTC, TAT, TGG, TGT, TTC\}$
$\{AAG, AAT, ACA, AGT, ATC, ATG, CAC, CAG, CCT, CGC, GAC, GAG, GCT, GGC, GTC, TAC, TGG, TGT, TTA, TTC\}$
$\{AAG, AAT, ACA, AGT, ATC, ATG, CAC, CAG, CCT, CGT, CTA, GAC, GAG, GCC, GGC, GTT, TAT, TGC, TGG, TTC\}$
$\{AAG, AAT, ACA, AGT, ATC, ATG, CAC, CAG, CCT, CGT, GAC, GAG, GCC, GGC, GTT, TAC, TGC, TGG, TTA, TTC\}$

$\{AAG, AAT, ACA, AGT, ATG, ATT, CAG, CAT, CCA, CGC, CTC, GAC, GAG, GCT, GGC, GTC, TAC, TGG, TGT, TTC\}$
$\{AAG, AAT, ACA, AGT, ATG, ATT, CAG, CAT, CCA, CGT, CTC, GAC, GAG, GCC, GGC, GTT, TAC, TGC, TGG, TTC\}$
$\{AAG, AAT, ACA, ATC, ATG, CAC, CAG, CCG, CGT, CTA, GAC, GAG, GCT, GGC, GTA, TAT, TCC, TGG, TGT, TTC\}$
$\{AAG, AAT, ACA, ATC, ATG, CAC, CAG, CCG, CGT, GAC, GAG, GCT, GGC, GTA, TAC, TCC, TGG, TGT, TTA, TTC\}$
$\{AAG, AAT, ACA, ATC, ATG, CAC, CAG, CCT, CGC, CTA, GAC, GAG, GCT, GGC, GTA, TAT, TCG, TGG, TGT, TTC\}$
$\{AAG, AAT, ACA, ATC, ATG, CAC, CAG, CCT, CGC, GAC, GAG, GCT, GGC, GTA, TAC, TCG, TGG, TGT, TTA, TTC\}$
$\{AAG, AAT, ACA, ATG, ATT, CAC, CAG, CCG, CGT, CTC, GAC, GAG, GCT, GGC, GTA, TAC, TCA, TGG, TGT, TTC\}$
$\{AAG, AAT, ACA, ATG, ATT, CAC, CAG, CCT, CGT, CTG, GAC, GAG, GCC, GGC, GTA, TAC, TCA, TGG, TGT, TTC\}$
$\{AAG, AAT, ACA, ATG, ATT, CAC, CAG, CCT, CGT, GAC, GAG, GCC, GGC, GTA, TAC, TCA, TGC, TGG, TTC, TTG\}$
$\{AAG, AAT, ACA, ATG, ATT, CAG, CAT, CCA, CGC, CTC, GAC, GAG, GCT, GGC, GTA, TAC, TCG, TGG, TGT, TTC\}$
$\{AAG, AAT, ACA, ATG, ATT, CAG, CAT, CCA, CGT, CTG, GAC, GAG, GCC, GGC, GTA, TAC, TCC, TGG, TGT, TTC\}$
$\{AAG, AAT, ACA, ATG, ATT, CAG, CAT, CCA, CGT, GAC, GAG, GCC, GGC, GTA, TAC, TCC, TGC, TGG, TTC, TTG\}$
$\{AAG, AAT, ACC, AGC, ATG, ATT, CAA, CAT, CCG, CGT, CTC, GAC, GAG, GCT, GGC, GTA, TAC, TGG, TGT, TTC\}$
$\{AAG, AAT, ACC, AGC, ATG, ATT, CAA, CAT, CCT, CGT, CTG, GAC, GAG, GCC, GGC, GTA, TAC, TGG, TGT, TTC\}$
$\{AAG, AAT, ACC, AGC, ATG, ATT, CAA, CAT, CCT, CGT, GAC, GAG, GCC, GGC, GTA, TAC, TGC, TGG, TTC, TTG\}$
$\{AAG, AAT, ACC, AGT, ATG, ATT, CAA, CAT, CCG, CGT, CTC, GAC, GAG, GCA, GGC, GTT, TAC, TGC, TGG, TTC\}$
$\{AAG, AAT, ACC, AGT, ATG, ATT, CAA, CAT, CCT, CGC, CTG, GAC, GAG, GCA, GGC, GTC, TAC, TGG, TGT, TTC\}$
$\{AAG, AAT, ACC, AGT, ATG, ATT, CAA, CAT, CCT, CGC, GAC, GAG, GCA, GGC, GTC, TAC, TGC, TGG, TTC, TTG\}$
$\{AAG, AAT, ACC, ATG, ATT, CAA, CAT, CCG, CGT, CTG, GAC, GAG, GCA, GGC, GTA, TAC, TCC, TGG, TGT, TTC\}$
$\{AAG, AAT, ACC, ATG, ATT, CAA, CAT, CCG, CGT, GAC, GAG, GCA, GGC, GTA, TAC, TCC, TGC, TGG, TTC, TTG\}$
$\{AAG, AAT, ACC, ATG, ATT, CAA, CAT, CCT, CGC, CTG, GAC, GAG, GCA, GGC, GTA, TAC, TCG, TGG, TGT, TTC\}$
$\{AAG, AAT, ACC, ATG, ATT, CAA, CAT, CCT, CGC, GAC, GAG, GCA, GGC, GTA, TAC, TCG, TGC, TGG, TTC, TTG\}$
$\{AAG, AAT, ACT, AGC, ATC, ATG, CAA, CAC, CCG, CGT, CTC, GAC, GAG, GCT, GGC, GTA, TAT, TGG, TGT, TTC\}$
$\{AAG, AAT, ACT, AGC, ATC, ATG, CAA, CAC, CCT, CGT, CTG, GAC, GAG, GCC, GGC, GTA, TAT, TGG, TGT, TTC\}$
$\{AAG, AAT, ACT, AGC, ATC, ATG, CAA, CAC, CCT, CGT, GAC, GAG, GCC, GGC, GTA, TAT, TGC, TGG, TTC, TTG\}$
$\{AAG, AAT, ACT, AGT, ATC, ATG, CAA, CAC, CCG, CGT, CTC, GAC, GAG, GCA, GGC, GTT, TAT, TGC, TGG, TTC\}$
$\{AAG, AAT, ACT, AGT, ATC, ATG, CAA, CAC, CCT, CGC, CTG, GAC, GAG, GCA, GGC, GTC, TAT, TGG, TGT, TTC\}$
$\{AAG, AAT, ACT, AGT, ATC, ATG, CAA, CAC, CCT, CGC, GAC, GAG, GCA, GGC, GTC, TAT, TGC, TGG, TTC, TTG\}$
$\{AAG, AAT, ACT, ATC, ATG, CAA, CAC, CCG, CGT, CTG, GAC, GAG, GCA, GGC, GTA, TAT, TCC, TGG, TGT, TTC\}$
$\{AAG, AAT, ACT, ATC, ATG, CAA, CAC, CCG, CGT, GAC, GAG, GCA, GGC, GTA, TAT, TCC, TGC, TGG, TTC, TTG\}$
$\{AAG, AAT, ACT, ATC, ATG, CAA, CAC, CCT, CGC, CTG, GAC, GAG, GCA, GGC, GTA, TAT, TCG, TGG, TGT, TTC\}$
$\{AAG, AAT, ACT, ATC, ATG, CAA, CAC, CCT, CGC, GAC, GAG, GCA, GGC, GTA, TAT, TCG, TGC, TGG, TTC, TTG\}$

Scaled down sequence set

The following list is a list of the used coding sequences in the experiments presented in chapter 5. The list contains only the FAST file commend of the sequences.

- >CCE57618 cds plasmid:HUSEC2011CHR1:pHUSEC2011-2:166:1143:1 gene: HUS2011 pII0001 gene biotype:protein coding transcript biotype:protein coding gene symbol:repA description:replication initiation protein RepFIB

- >CCE57621 cds plasmid:HUSEC2011CHR1:pHUSEC2011-2:2531:2692:-1 gene: HUS2011 pII0004 gene biotype:protein coding transcript biotype:protein coding description:plasmid stabilisation system family protein
- >CCE57622 cds plasmid:HUSEC2011CHR1:pHUSEC2011-2:2894:3076:-1 gene: HUS2011 pII0005 gene biotype:protein coding transcript biotype:protein coding description:putative transcriptional regulator
- >CCE57623 cds plasmid:HUSEC2011CHR1:pHUSEC2011-2:3145:3420:-1 gene: HUS2011 pII0006 gene biotype:protein coding transcript biotype:protein coding description:plasmid stabilisation system family protein
- >CCE57626 cds plasmid:HUSEC2011CHR1:pHUSEC2011-2:5252:5452:-1 gene: HUS2011 pII0009 gene biotype:protein coding transcript biotype:protein coding description:IncFII RepA protein, truncated
- >CCE57627 cds plasmid:HUSEC2011CHR1:pHUSEC2011-2:5745:5999:-1 gene: HUS2011 pII0010 gene biotype:protein coding transcript biotype:protein coding gene symbol:repB description:replication regulatory protein
- >ENA—AFU90012—AFU90012.1 Abalone herpesvirus Victoria/AUS/2009 hypothetical protein
- >ENA—AFU90013—AFU90013.1 Abalone herpesvirus Victoria/AUS/2009 putative eukaryotic translation initiation factor
- >ENA—AFU90014—AFU90014.1 Abalone herpesvirus Victoria/AUS/2009 hypothetical protein
- >CCDS3.1—Hs109—chr1
- >CCDS8138.1—Hs109—chr11
- >CCDS3822.1—Hs109—chr4
- >ENA—ASP44138—ASP44138.1 Bat SARS coronavirus HKU3 partial RNA-dependent RNA polymerase
- >ENA—ASP44139—ASP44139.1 Bat SARS coronavirus HKU3 partial RNA-dependent RNA polymerase
- >ENA—ASP44140—ASP44140.1 Bat SARS coronavirus HKU3 partial RNA-dependent RNA polymerase

- >YAL001C TFC3 SGDID:S000000001, Chr I from 151006-147594,151166-151097, Genome Release 64-2-1, reverse complement, intron sequence removed, Verified ORF, "Subunit of RNA polymerase III transcription initiation factor complex; part of the TauB domain of TFIIIC that binds DNA at the BoxB promoter sites of tRNA and similar genes; cooperates with Tfc6p in DNA binding; largest of six subunits of the RNA polymerase III transcription initiation factor complex (TFIIIC)"
- >YAL002W VPS8 SGDID:S000000002, Chr I from 143707-147531, Genome Release 64-2-1, Verified ORF, "Membrane-binding component of the CORVET complex; involved in endosomal vesicle tethering and fusion in the endosome to vacuole protein targeting pathway; interacts with Vps21p; contains RING finger motif"
- >ENA—AAA03517—AAA03517.1 Caenorhabditis elegans kinesin-related protein
- >ENA—AAA03544—AAA03544.1 Caenorhabditis elegans BMP receptor
- >CCE57640 cds plasmid:HUSEC2011CHR1:pHUSEC2011-2:13466:17179:-1 gene:HUS2011 pII0023 gene biotype:protein coding transcript biotype:protein coding description: conjugative transfer DNA-nicking and unwinding protein TraI, truncated
- >CCE57648 cds plasmid:HUSEC2011CHR1:pHUSEC2011-2:20958:21758:-1 gene: HUS2011 pII0031 gene biotype:protein coding transcript biotype:protein coding description:hypothetical protein
- >YGR119C NUP57 SGDID:S000003351, Chr VII from 729671-728046, Genome Release 64-2-1, reverse complement, Verified ORF, "FG-nucleoporin component of central core of the nuclear pore complex; contributes directly to nucleocytoplasmic transport and maintenance of the nuclear pore complex (NPC) permeability barrier; found in stable complex with Nic96p and two other FG-nucleoproteins (Nsp1p and Nup49p)"

List of Figures

List of Tables

Bibliography

[1] D. Arquès and C. Michel. A complementary circular code in the protein coding genes. *J. Theor. Biol.*, 182(1):45–58, September 1996.

[2] A. Ball and L. Cummings. Extremal digraphs and comma-free codes. *Ars Combinatoria*, 1(1):239–251, 1976.

[3] P. Baranov, M. Venin, and G. Provan. Codon size reduction as the origin of the triplet genetic code. *PLOS ONE*, 4(5):1–9, 2009.

[4] E. Bell, P. Boehnke, T. Harrison, and W. Mao. Potentially biogenic carbon preserved in a 4.1 billion-year-old zircon. *Proceedings of the National Academy of Sciences*, 112(47):14518–14521, November 2015.

[5] J. Berstel, D. Perrin, and C. Reutenauer. *Codes and automata*. by Jean Berstel, Dominique Perrin and Christophe Reutenauer, 2005.

[6] J. Bowman, N. Hud, and L. Williams. The Ribosome Challenge to the RNA World. *Journal of Molecular Evolution*, 80(3):143–161, April 2015.

[7] P. Błażej, D. Kowalski, D. Mackiewicz, M. Wnetrzak, D. Aloqalaa, and P. Mackiewicz. The structure of the genetic code as an optimal graph clustering problem. preprint, Systems Biology, May 2018.

[8] S. Chatterjee and S. Yadav. The Origin of Prebiotic Information System in the Peptide/RNA World: A Simulation Model of the Evolution of Translation and the Genetic Code. *Life*, 9(1):25, March 2019.

[9] D. Cisowski. Tessera-based encoding of the mitochondrial genome. *Bachelor-Thesis, Mannheim*, 2015.

[10] C. Clarke. Hypothetical stages in the mutational evolution of the present-day genetic code from its rny, comma-free ancestor. *Journal of Theoretical Biology*, 99(2):397–403, November 1982.

[11] F. Crick. Origin of the Genetic Code. *Nature*, 213(5072):119–119, January 1967.

[12] F. Crick, J. Griffith, and L. Orgel. Codes without commas. *Proc. Natl. Acad. Sci. U.S.A.*, 43(5):416–421, 1957.

[13] S. Crick, F.and Brenner, A. Klug, and G. Pieczenik. A speculation on the origin of protein synthesis. *Origins of Life*, 7(4):389–397, December 1976.

[14] C. Darwin. *On the origin of species*. D. Appleton and Co.,, 1871.

[15] G. Dila, C. Michel, O. Poch, R. Ripp, and J. Thompson. Evolutionary conservation and functional implications of circular code motifs in eukaryotic genomes. *Biosystems*, 175:57–74, 2019.

[16] J. Dolezel, J. Bartos, H. Voglmayr, and J. Greilhuber. Nuclear DNA content and genome size of trout and human. *Cytometry. Part A: The Journal of the International Society for Analytical Cytology*, 51(2):127–128; author reply 129, February 2003.

[17] K. El Soufi and C. Michel. Circular code motifs near the ribosome decoding center. *Computational Biology and Chemistry*, 59:158–176, December 2014.

[18] Karim El Soufi. *Study of circular code motifs in nucleic acid sequences*. Theses, Université de Strasbourg, January 2017.

[19] E. Fimmel, A. Danielli, and L. Strüngmann. On dichotomic classes and bijections of the genetic code. *Journal of Theoretical Biology*, 336:221–230, November 2013.

[20] E. Fimmel, S. Giannerini, D. Gonzalez, and L. Strüngmann. Circular codes, symmetries and transformations. *Journal of Mathematical Biology*, 70(7):1623–1644, 2015.

[21] E. Fimmel, S. Giannerini, D. Gonzalez, and L. Strüngmann. Dinucleotide circular codes and bijective transformations. *Journal of Theoretical Biology*, 386:159–165, December 2015.

[22] E. Fimmel, C. Michel, F. Pirot, J. Sereni, M. Starman, and L. Strüngmann. The relation between k-circularity and circularity of codes. *Bulletin of Mathematical Biology*, 82(8):105, August 2020.

[23] E. Fimmel, C. Michel, F. Pirot, J. Sereni, and L. Strüngmann. Comma-free codes over finite alphabets. working paper or preprint, November 2019.

[24] E. Fimmel, C. Michel, M. Starman, and L. Strüngmann. Self-complementary circular codes in coding theory. *Theory Biosci.*, 137(1):51–65, April 2018.

[25] E. Fimmel, C. Michel, and L. Strüngmann. Diletter circular codes over finite alphabets. *Mathematical Biosciences*, 294:120–129, 2017.

[26] E. Fimmel, C. Michel, and L. Strüngmann. Strong comma-free codes in genetic information. *Bulletin of Mathematical Biology*, 79:1796–1819, 2017.

[27] E. Fimmel, C. Michel, and L. Strüngmann. n-nucleotide circular codes in graph theory. *Phil. Trans. R. Soc. A*, 374(2063):20150058, 2016.

[28] E. Fimmel, M. Starman, and L. Strüngmann. Circular tessera codes in the evolution of the genetic code. *Bulletin of Mathematical Biology*, 82(4):48, April 2020.

[29] E. Fimmel and L. Strüngmann. On the hierarchy of trinucleotide n-circular codes and their corresponding amino acids. *Journal of Theoretical Biology*, 364:113–120, 2015.

[30] E. Fimmel and L. Strüngmann. Maximal dinucleotide comma-free codes. *Journal of Theoretical Biology*, 389:206–213, 2016.

[31] E. Fimmel and L. Strüngmann. Yury borisovich rumer and his 'biological papers' on the genetic code. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2063):20150228, March 2016.

[32] E. Fimmel and L. Strüngmann. Linear codes and the mitochondrial genetic code. *Biosystems*, 184:103990, October 2019.

[33] P. Forterre, S. Gribaldo, and C. Brochier. Luca : À la recherche du plus proche ancêtre commun universel. *médecine/sciences*, 21(10):860–865, October 2005.

[34] The R Foundation. R:what is r? `https://www.r-project.org/about.html`. Accessed: July 29, 2020.

[35] S. Freeland and L. Hurst. The genetic code is one in a million. *Journal of Molecular Evolution*, 47:238–248, 1998.

[36] S. Golomb, L. Welch, and M. Delbrück. *Construction and properties of comma-free codes.* i kommission hos Ejnar Munksgaard, 1958.

[37] D. Gonzalez. The mathematical structure of the genetic code. In Marcello Barbieri and Jesper Hoffmeyer, editors, *The Codes of Life: The Rules of Macroevolution*, pages 111–152. Springer Netherlands, Dordrecht, 2008.

[38] D. Gonzalez. The mathematical structure of the genetic code. In M. Barbieri and J. Hoffmeyer, editors, *The Codes of Life: The Rules of Macroevolution*, pages 111–152. Springer Netherlands, Dordrecht, 2008.

[39] D. Gonzalez, S. Giannerini, and R. Rosa. On the origin of the mitochondrial genetic code: Towards a unified mathematical framework for the management of genetic information. *Nature Precedings*, page npre.2012.7136.1, April 2012.

[40] D. Gonzalez, S. Giannerini, and R. Rosa. On the origin of degeneracy in the genetic code. *Interface Focus*, 9(6):20190038, December 2019.

[41] M. Gumbel, E. Fimmel, A. Danielli, and L. Strüngmann. On models of the genetic code generated by binary dichotomic algorithms. *Bio Systems*, 128:9–18, February 2015.

[42] M. Gumbel, K. Kristian, and M. Starman. Genetic code analysis toolkit (gcat). `https://www.cammbio.hs-mannheim.de/research/software/gcat.html`, 2016. Accessed: July 29, 2020.

[43] M. Gumbel and P. Wiedemann. Motif lengths of circular codes in coding sequences. submitted, 2020.

[44] P. Higgs. A four-column theory for the origin of the genetic code: tracing the evolutionary pathways that gave rise to an optimized code. *Biology Direct*, 4(1):16, 2009.

[45] K. Ikehara. Origins of gene, genetic code, protein and life: comprehensive view of life systems from a GNC-SNS primitive genetic code hypothesis. *Journal of Biosciences*, 27(2):165–186, March 2002.

[46] E. Koonin. Frozen accident pushing 50: stereochemistry, expansion, and chance in the evolution of the genetic code. *Life*, 7:1–13, 2017.

[47] E. Koonin and A. Novozhilov. Origin and evolution of the genetic code: the universal enigma. *IUBMB life*, 61(2):99–111, February 2009.

[48] K. Leeuw and J. Bergstra, editors. *The history of information security: a comprehensive handbook*. Elsevier, Amsterdam ; London, 2007. OCLC: ocm85897846.

[49] G. Löffler. *Basiswissen Biochemie mit Pathobiochemie*. Springer, 2008. OCLC: 723880951.

[50] C. Michel. Circular code motifs in transfer and 16sribosomal rnas: a possible translation code in genes. *Computational biology and chemistry*, 37:24–37, 2012.

[51] C. Michel. Circular code motifs in transfer rnas. *Computational biology and chemistry*, 45:17–29, 2013.

[52] C. Michel. A genetic scale of reading frame coding. *Computational biology and chemistry*, 355:83–94, 2014.

[53] C. Michel. The maximal c^3 self-complementary trinucleotide circular code x in genes of bacteria, eukaryotes, plasmids and viruses. *Journal of Theoretical Biology*, 380:156–177, 2015.

[54] C. Michel. The maximal c^3 self-complementary trinucleotide circular code x in genes of bacteria, archaea, eukaryotes, plasmids and viruses. *Life (Basel)*, 7(2), April 2017.

[55] C. Michel, V. Nguefack Ngoune, O. Poch, R. Ripp, and J. Thompson. Enrichment of circular code motifs in the genes of the yeast saccharomyces cerevisiae. *Life*, 7(52):1–20, 2017.

[56] C. Michel and G. Pirillo. Identification of all trinucleotide circular codes. *Computational Biology and Chemistry*, 34:122–125, 2010.

[57] C. Michel and G. Pirillo. A permuted set of a trinucleotide circular code coding the 20 amino acids in variant nuclear codes. *Journal of Theoretical Biology*, 319:116–121, 2013.

[58] C. Michel, G. Pirillo, and M. Pirillo. A relation between trinucleotide comma-free codes and trinucleotide circular codes. *Theoretical Computer Science*, 401:17–26, 2008.

[59] B. Neveln. Comma-free and synchronizable codes. *Journal of Theoretical Biology*, 144(2):209–212, May 1990.

[60] M. Nirenberg and J. Matthaei. The dependence of cell-free protein synthesis in E. coli upon naturally occurring or synthetic polyribonucleotides. *Proceedings of the National Academy of Sciences*, 47(10):1588–1602, October 1961.

[61] A. Novozhilov, Y. Wolf, and E. Koonin. Evolution of the genetic code: partial optimization of a random code for robustness to translation error in a rugged fitness landscape. *Biology Direct*, 2(1):24, 2007.

[62] J. Paps and P. Holland. Reconstruction of the ancestral metazoan genome reveals an increase in genomic novelty. *Nature Communications*, 9(1):1730, December 2018.

[63] A. Patel. The triplet genetic code had a doublet predecessor. *Journal of Theoretical Biology*, 2004.

[64] S. Pelc and M.G. Welton. Stereochemical relationship between coding triplets and amino-acids. *Nature*, 209:868–870, 1966.

[65] B. Piette and J. Heddle. A Peptide–Nucleic Acid Replicator Origin for Life. *Trends in Ecology & Evolution*, 35(5):397–406, May 2020.

[66] Cambridge University Press. Dictionary cambridge: Codes. `https://dictionary.cambridge.org/de/worterbuch/englisch/code`, 2014. Accessed: October 05, 2020.

[67] Yu. Rumer. Translation of 'systematization of codons in the genetic code [i]' by yu. b. rumer (1966). *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2063):20150446, March 2016.

[68] H. Seligmann. Putative anticodons in mitochondrial trna sidearm loops: Pocketknife trnas? *Journal of Theoretical Biology*, 340:155–163, January 2014.

[69] J. Shepherd. Method to determine the reading frame of a protein from the purine/pyrimidine genome sequence and its possible evolutionary justification. *Proceedings of the National Academy of Sciences*, 78(3):1596–1600, March 1981.

[70] K. Soufi and C. Michel. Circular code motifs in genomes of eukaryotes. *Journal of Theoretical Biology*, 408:198–212, 2016.

[71] M. Starman. Genetic code analysis toolkit in r (gcatr). `https://github.com/StarmanMartin/GCATR`, 2019. Accessed: July 29, 2020.

[72] E. Szathmáry. Four letters in the genetic alphabet: a frozen evolutionary optimum? *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 245(1313):91–99, August 1991.

[73] D. Theobald. A formal test of the theory of universal common ancestry. *Nature*, 465(7295):219–222, May 2010.

[74] K. Watanabe and S. Yokobori. How the early genetic code was established?: Inference from the analysis of extant animal mitochondrial decoding systems. In V. Erdmann, W. Markiewicz, and J. Barciszewski, editors, *Chemical Biology of Nucleic Acids: Fundamentals and Clinical Applications*, pages 25–40. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[75] T. Wilhelm and S. Nikolajewa. A new classification scheme of the genetic code. *Journal of Molecular Evolution*, 59(5):598–605, November 2004.

[76] C. Woese. Order in the genetic code. *Proceedings of the National Academy of Sciences U.S.A.*, 54:71–75, 1965.

[77] J. Wong. A co-evolution theory of the genetic code. *Proceedings of the National Academy of Sciences U.S.A.*, 72:1909–1912, 1975.

[78] H. Wu, S. Bagby, and J. van den Elsen. Evolution of the genetic triplet code via two types of doublet codons. *Journal of molecular evolution*, 61:54–64, 2005.

[79] M. Yarus. The genetic code and rna-amino acid affinities. *Life*, 7:1–16, 2017.