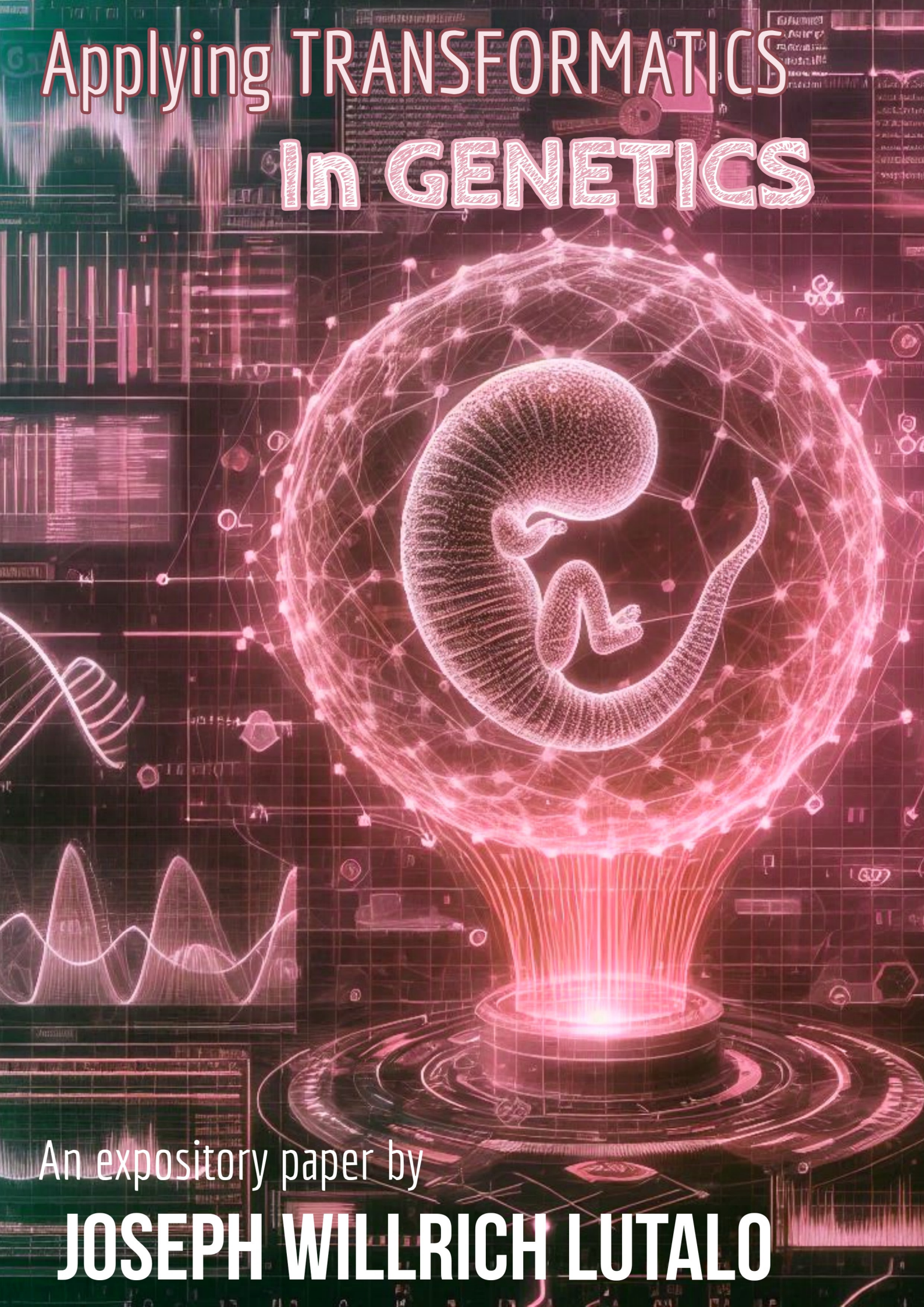
Applying TRANSFORMATICS In GENETICS



An expository paper by

JOSEPH WILLRICH LUTALO

Applying **TRANSFORMATICS** in GENETICS

Willrich J. Lutalo*
joewillrich@gmail.com, jwl@nuchwezi.com

August 7, 2025

Abstract

Though still just a paper, this work brings to surface the importance of leveraging the mathematical statistical theory recently named "Transformatics", that deals with the study, processing and analysis of especially ordered sequences of symbols. It has been demonstrated to be a credible theory in designing, specifying or explaining the properties of automata operating on sequences to produce other sequences — so-called sequence transformers. So, in this particular work, we take that body of knowledge, as well as what we know about the critically important science of how biological life-forms get to be defined, transformed and expressed in nature via the special genetic code known as DNA (deoxyribonucleic acid), that is best modeled conceptually as an ordered sequence of genes or more technically, a sequence of special combinations of any of four chemical bases (amino acids) or just "nucleobases" — Adenine(A), Cytosine(C), Guanine(G), and Thymine(T), into a finite sequence typically expressed as a double-helix ladder structure, that can then be decoded by convenient biological mechanisms so as to express or rather, manufacture one or more essential life-building compounds such as proteins that underlie the synthesis of specialized aspects of the organism's body such as skin, bone and muscle tissue in an animals or into cell-wall tissue, photosynthesis machinery and others, for plant organisms. This work demonstrates how several ideas first compiled under the transformatics umbrella can be well applied in problems relating to general genetics; we for example see how to quantify how far apart or different any two organisms might be based on the anagram distance between their genetic codes, this likewise being applicable to also sub-sequences of DNA that might underlie the expression of just particular biological machines or mechanisms. We also consider how to leverage the concept of the modal sequence statistic in analyzing not just how similar different DNA sequences might be in terms of their relative composition of the basic nucleobases, but also in terms of their relative composition at higher abstraction levels such as at the level of amino-acids or large n-gram subsequences. We finally consider the matter of how, by leveraging the idea that a modal sequence encodes a summary about some larger sequence or an entire population of them, that we can then approach DNA as though it were a special statistical summary just like the MSS from transformatics theory, and then like how complex sequences could be constructed from summary statistics via certain protraction and multiplication transformers in earlier work, we use a thought experiment to demonstrate how DNA would be transcribed into both an mRNA-like structure and then which can be further transcribed into actual body structures that allow the organism to occupy space and appear or behave in a particular way. Though this work is mostly theoretical, we anticipate that this discussion and exposition shall inspire actual domain experts and other researchers interested in genetics, genetic engineering and other sciences to pick up and apply our transformatics theory and ideas in both theory and practice.

*Currently a volunteering & Independent Researcher at Nuchwezi Research — https://nuchwezi.com

Keywords: Applied Mathematical Statistics, Transformatics, Artificial Statistical Intelligence, Information Processing, Ordered Sequences, DNA sequences, Genetic Code, Genetic Analysis

> Sex seems to have been invented around two billion years ago. Before then, new varieties of organisms could arise only from the accumulation of random mutations --- the selection of changes, letter by letter, in the genetic instructions. Evolution must have been agonizingly slow. With the invention of sex, two organisms could exchange whole paragraphs, pages and books of their DNA code, producing new varieties ready for the sieve of selection. Organisms are selected to engage in sex --- the ones that find it uninteresting quickly become extinct. And this is true not only for the microbes of two billion years ago. We humans have a palpable devotion to exchanging segments of DNA today.
>
> — **Carl Sagan**, *COSMOS*, 1981[1]

1 Introduction

> The material basis of heredity is DNA, a ladder-like molecule which carries a message in the form of a 'four-letter' code, the letters being four chemical bases, each of which may occupy any rung in the ladder.
>
> — **The Oxford Companion to the Mind**[2]

In reality, we find that living, real organisms are influenced by genetics, their environment, bits of randomness and sometimes emergent behaviors that might not be readily captured by strict rules such as the genetic code of life. However, away from all possibilities, and focusing on what can be said of life expression via the genetic code known as DNA (the *deoxyribonucleic acid*) or RNA (ribonucleic acid), especially when applied to the vast spectrum of natural organisms on earth — from basic **prokaryotes** (single-celled organisms) such as the simple bacteria that actually have no nucleus[3], to **eukaryotes**; all the way from basic one-cell kinds such as amoeba[4] all the way to sophisticated creatures such as oak trees, vultures, dolphins and human beings! Talking of which, it might be important to set clear that though **viruses** are a kind of life-form[5], and yet, they are neither prokaryotes nor eukaryotes — especially because, fundamentally, they are merely some genetic code (DNA or RNA) "enclosed in a protein coat, lacking cell membranes or organelles"[5] — more technically they are **acellular entities**.

So, if we bring on-board ideas from **transformatics**[6] — a new mathematical statistics theory dealing with sequences of symbols or those of named structures and their processing as well as analysis, for example, if we take the concept

of leveraging statistical measures to summarize essential properties of sequences such as DNA — say, with use of the modal sequence statistic (MSS), we find that, independently of, and without needing to first consult or worry about mainstream genetic code analysis or genetic engineering theory and mechanics, that we can say many useful things about genetic code sequences and that we might be able to break new ground or solve some otherwise still intractable problems concerning sequences of genetic code.

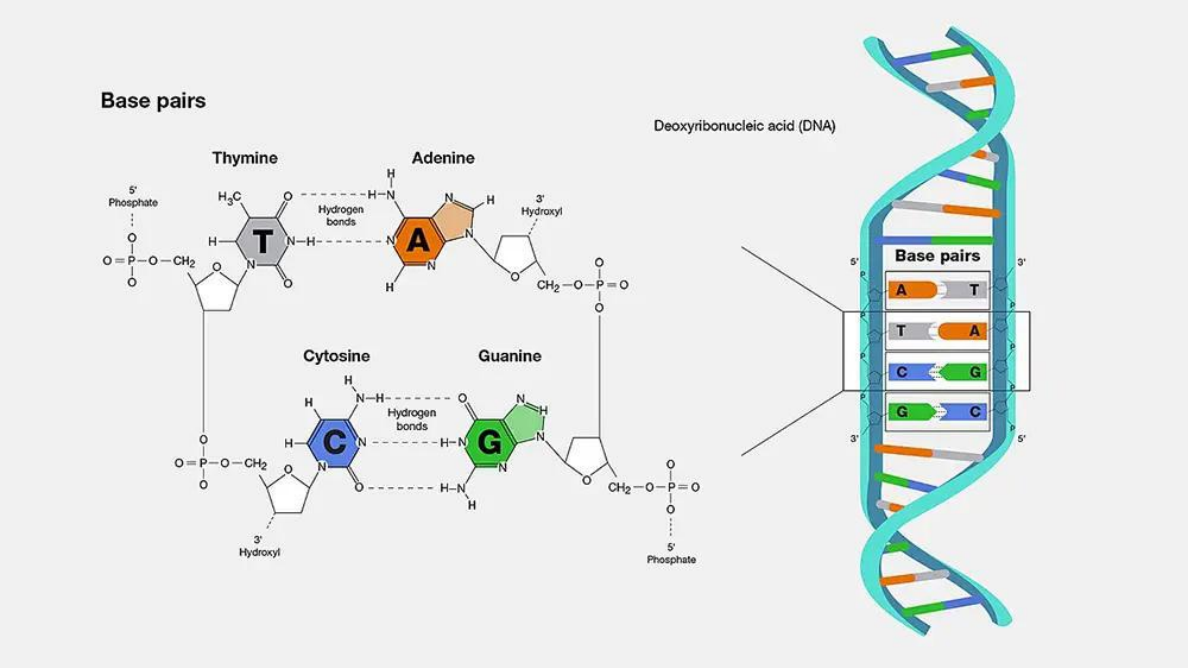


Figure 1: A basic diagram of the chemical structure of DNA base-pairs from [7]

For example, by borrowing a useful DNA-as-library metaphor from Venville et al[8], we would come to appreciate that by looking at *DNA as a library of books*, we have a model such as:

1. **Library**: DNA (as a whole) — the **Genome**, as the complete collection of genomic data about an organism.
2. **Bookshelves**: **Chromosomes** as organized storage of genes, and that they (chromosomes) are long, coiled-up strands of pairs of genes (essentially, the chromosome is a combination of two strands of genes, with one called the *template* and the other a *complement*, and that when they come together to form the "ladder" structure, at each step, the two genes forming a step are paired such that $A \leftrightarrow T$ and $C \leftrightarrow G$ [9] — also see **Figure 1**). So, for example, humans have 46 chromosomes in total in their "DNA library" (23 chromosome pairs, consisting of 22 autosomal pairs plus one pair of sex chromosomes). We know that during reproduction, the [full] genome (46 chromosomes in humans) splits up by half in either parent (via *meiosis*, which somewhat shuffles

the complete genome and then halves it[10]), so that only one-half of the otherwise well-paired template-complement set that is the chromosome strand from each parent (as either sperm or ovum) goes to contribute to the final chromosome collection-set of the offspring[11][2]. Whereas, after fertilization or during normal cell division, the entire chromosome set (with well-paired strands) is duplicated/replicated wholesomely and losslessly so that the second/new cell thus created has an exact copy of the same chromosome set (entire genome) as was in the source cell[12].

3. **Books**: The **Genes** which make up a chromosome are the books in our genome library. Each gene is essentially a collection of "words" that are a sequence of one or more codons (see below), and which taken together, contain enough information/instructions to specify how to produce a specific protein[2][8].

4. **Words**: And then, under genes, we have **Codons** that are like "words" in each book, and each codon **only** codes for a single **amino acid**. Basically, a codon is a combination of any 3 of the "letters" of genetic code, for which, there are exactly four for DNA (A, C, G, T) and four for RNA (A, C, G, U). We also know that there are at most, 64 possible unique combinations of the four letters into triplets/3-grams/the codons[2].

5. **Letters**: Finally, at the most basic level of our genetic code/DNA-library/genome, we have "letters" that are technically known as **nucleotides**, and the nucleotide essentially each contains **a single nucleobase**, with only one difference between DNA and RNA as such; for DNA: $\{A, C, G, T\}$, and for RNA: $\{A, C, G, U\}$ — we have seen the names of the four letters for DNA, and the new one for RNA, 'U', stands for "Uracil".

So, with that introduction clearing up much of the basic genetics nomenclature and concepts that we shall use in the rest of this work, we then dive into the meat of our undertaking as such; **Section 2** shall introduce the use of terminology and notation from transformatics to describe facts about genome sequences at various levels of abstraction. We shall look at using sequence symbol sets, sequence abstraction using sub-sequence symbols that can later be re-transformed into flat sequences, the idea of sequence cardinality when applied to DNA sequences and more. Then in **Section 4** we shall deepen our discussion by considering how some of the measures from transformatics might be applied in genetic sequence analysis. We shall look into the ADM and PCR especially — the other for cases where any two sequences are of the same length and similar symbol sets, and the other for cases where these need not be the same across the sequences under analysis. Then in **Section 8** we shall take on the interesting matter for how, despite being merely a sequence of symbols, genetic code sequences actually can be likened to

how a MSS can be used to specify how to reconstruct some other sequence. First, in **Section 9** we shall first look at an overview of how actual interpretation or execution (more conventionally just referred to as "gene translation") results in the manufacturing of proteins, and shall likewise look at some example actual gene translation in general and with a particular case. In **Section 10** we shall then dive into a more hands-on exploration of this matter, with a hypothetical genome system (the **Numero-Gene Code**) that can allow us to not only creatively explore what genetic code is about, but which can also allow us to approach the rather complex matter of how DNA gets interpreted/translated into actual living tissue as well as a complete living organism. We shall introduce some two systems for how to decode our numero-gene DNA code into a kind of mRNA via the **Ozin-Transformer** and from ozin-gene code into actual tissue/proteins via the **Plato-Form Generator** that allows us to transcribe DNA into an orgnanism that has a predictable characteristic appearance and geometry, but also which still contains its essential genetic code just like normal living organisms do. Then we shall wrap-up in **Section 11**, looking at what we have accomplished, what remains to be done, and what the implications of this undertaking might be.

2 Sequence Symbol Sets and Sequence Transforms Applied to Genetic Code

> **NOTE:**
> On **Potentially Similar Mathematics**: *Frobenoids?*
>
> Frobenioids[13], as introduced by **Prof. Shinichi Mochizuki**, offer a category-theoretic abstraction of arithmetic structures such as divisors and line bundles, encoding transformations via morphisms that resemble symbolic rewritings under algebraic constraints[14]. In spirit, they parallel the logic of sequence transformers (from transformatics[6]), which operate by mapping one symbolic sequence to another through rule-based transformations constrained by set-theoretic, statistical or general mathematical logic. However, even though both systems might serve as structured environments for encoding and navigating symbolic changes across mathematical objects such as sequences and/or sets, we just wanted to bring this up here, so students and researchers interested in our transformatics mathematics, and who wish to take the discussions and ideas we present in this work to even more advanced levels or in unconventional directions, pick a leaf and comfortably proceed to do so.

For the purposes of appreciating transformatics from a genetics engineering or general genetics research perspective, it shall be important to note that like in the original transformatics paper[6], beginning by appreciating that whatever formalisms and mechanics we might develop or talk about concerning genetic code or rather DNA, had better begin with an appreciation that we can model DNA as merely an ordered sequence of symbols.

In the introduction (see **Section 1**), we have already called out both the names

and symbols assigned to the most basic units of any genetic code; essentially, the *nucleobases* (or rather, nucleotides). For purposes of simplifying our mathematical logic later on, we shall here neatly define what roles these units play in the grand scheme concerning DNA and RNA code. Basically, we shall want to define the symbol sets for any DNA sequence and the symbol set for any RNA sequence.

Definition 1 (The **DNA Symbol Set**, ψ_{DNA})**.** *For any possible sequence of standard deoxyribonucleic acid (**DNA**) for any possible living organism, the distinct nucleic acid base units are known as nucleotides[15], and these are essentially and exactly only four*[a]*;*

- *Adenine(A)*
- *Cytosine(C)*
- *Guanine(G)*
- *Thymine(T)*

And these are mapped to their representative, distinct single-letter symbols as shown. Thus, any possible DNA sequence must always consist of only one or more of those elements and nothing else. Thus, we might sum this up, using the symbol set concept[16] as applied to sequences[6] as such:

$$\psi(DNA) = \{A, T, C, G\} \tag{1}$$

***Equation 1** helps to appreciate the extra non-intuitive fact that the special ordering of DNA base symbols in the order A-T-C-G is what is conventionally accepted[17][15] or commonly found in most genetics literature*[b]*.*
*However, and especially because, for transformatics, we wish to work with an **ordered symbol set**[18] and not just any possible symbol set so that we can apply mathematical logic that respects the ordering of terms in any ordered sequence[6], we shall then assume a convention similar to how we might derive an ordered symbol set for a sequence of numbers in some base (the concept $\psi_\beta(\Theta)$ — see **Definition 5** in [18]), and given we are using Latin-Alphabet symbols (from ψ_{az}[6]) for ψ_{DNA}, we might as well better define the **Lexically Ordered DNA Symbol Set**, ψ_{DNA} as such:*

$$\psi_{DNA} = \psi_{az}(DNA) = \langle A, C, G, T \rangle \tag{2}$$

*For all practical purposes unless where we merely wish to emphasize adherence to the tradition of ordering the nucleotides by their pairing order, we shall essentially imply $\psi_{az}(DNA)$ or rather ψ_{DNA} when we talk of the **DNA Symbol Set**.*

[a] Actually, or rather, in general, for nucleic acid sequences, the bases are four for either DNA or RNA, but, there are also conventions that extend this set to 17 or more to cater for cases like where there might be ambiguity about what the exact nucleotide in a particular position might be[15].

[b] It shall be important to bring it out at this point, that, especially for non-domain experts — people not trained in or normally practicing in genetics or related fields, that the common ordering of the nucleotides in the A-T-C-G ordering might seem unconventional or peculiar! For example, one might wonder, why are they not listed in their alphabetical order? So, for purposes of settings things clear for everyone, the author consulted a reliable research assistant on this matter[14], and it was made clear that: "The order **A, T, C, G** isn't alphabetical, and yet it's the most commonly used sequence when referring to DNA bases." We further learn that, the order A-T-G-C reflects a mix of historical usage and bio-chemical structure; **base-pairing logic**: that the DNA's double-helix is stabilized by "complementary base pairing" in which A pairs with T (via 2 hydrogen bonds), and C pairs with G (via 3 hydrogen bonds), so that listing A with its partner T and then C with G emphasizes this pairing symmetry[14]. Further, we learn that early molecular biology texts and sequencing protocols (especially post-Watson & Crick, 1953) adopted this order to reflect the "functional relationships" between bases. It became entrenched in educational materials, sequencing software, and databases. And lastly, that in visual and structural conventions — such as in diagrams (see **Figure 1**) and models, A–T and C–G are often shown side-by-side. Listing them in this order reinforces the **duality** of the DNA ladder's rungs[14]. Lastly, that though there is no single documented moment when this convention begun, that the A-T-C-G order likely solidified in the **1970s - 1980s** during the rise of **Sanger sequencing** (developed in the 1970s), GenBank and EMBL databases, and overall in textbooks and molecular biology curricula.

Now that we have ψ_{DNA} and ψ_{RNA} well defined, we might immediately apply them to their finest use here: defining and supporting the special symbol set, ψ_{na}, that would or could allow for several kinds of nucleic acid sequences — such as DNA for code stored in chromosomes and mitochondria, various kinds of RNA

(mRNA, tRNA,...) and even synthetic/artificial/conceptual and also **random nucleic acid sequences**, etc. using a single universal nucleic acid symbol set. Thus we define ψ_{na} below:

Definition 2 (The ***Universal Nucleic Acid Symbol Set***, ψ_{na})**.** *Any nucleic acid sequence, Θ, obeys the following law:*

Law 1 (**Nucleic Acid Identifier Set**: ψ_{na})**.** *The universal symbol set ψ_{na} spans any natural nucleic acid sequence Θ.*

Proof. $\psi_{na} = \psi_{DNA} \cup \psi_{RNA} = \langle A, C, G, T, U \rangle$ □

As with many kinds of sequences dealt with in transformatics, the possession of a particular symbol set, such as ψ_{na}, allows for the practical use of those sets to implement logic systems that can operate on signal based on symbolic expressions of well ordered elements in sequences or sub-sequences. For this matter then, we shall likewise want to make formal, the concept of a **na-Sequence**:

Definition 3 (The $na-$**Sequences** — (D/R)-NA Sequences: $\Theta_{na} : \mathbb{N} \times \psi_{na}$)**.** *Any sequence of DNA or RNA nucleotides — basically a sequence of nucleic acids, expressible as some sequence of symbols from ψ_{na} is a **na-Sequence** and its symbol set is a superset of both ψ_{DNA} and ψ_{RNA}. Equivalently:*

$$\forall \Theta_{na} = \langle a_i, \rangle, \quad a_i \in \psi_{na} \implies a_i \in \psi_{DNA} \vee a_i \in \psi_{RNA} \tag{3}$$

If it is not immediately clear what the significance of **Defition 1** is or why it's important to unambiguously define ψ_{DNA}, then perhaps the mathematical discussions in later sections like **Section 4** and **Section 8** might this more obvious. That said, since we know that nucleic acid sequences come in two flavors[15], and since we have covered the essential ground for DNA symbol sets, we need now also consider the ordered RNA symbol set.

Definition 4 (The **RNA Symbol Set**, ψ_{RNA})**.** *For any possible sequence of standard ribonucleic acid (**RNA**) for any possible living organism, the distinct nucleic acid base units are known as nucleotides[15], and these are essentially and exactly only four;*

- *Adenine(A)*
- *Cytosine(C)*
- *Guanine(G)*
- *Uracil(U)*

And these are mapped to their representative, distinct single-letter symbols as shown. Thus, any possible RNA sequence must always consist of only one or more of those elements and nothing else. Thus, we might sum this up, using the symbol set concept[16] as applied to sequences[6] as such:

$$\psi(RNA) = \{A, U, C, G\} \tag{4}$$

***Equation 4** helps to appreciate the traditional ordering of RNA base symbols in the order A-U-C-G reminiscent of the natural base-pairing order of DNA after it is translated into RNA (U merely replacing T)[19]. And as with DNA, we shall want to have a proper, meaningful **ordered symbol set**[18] for the RNA base symbols that we can later use in mathematical logic. Thus we shall equivalently define one for any standard RNA sequences as:*

$$\psi_{RNA} = \psi_{az}(RNA) = \langle A, C, G, U \rangle \tag{5}$$

*And for all practical purposes in this work as well as after, we shall essentially imply $\psi_{az}(RNA)$ or rather ψ_{RNA} when we talk of the **RNA Symbol Set** or the **Lexically Ordered RNA Symbol Set**.*

So, with those very essential definitions out of the way, we can begin to think of how we might appreciate and apply concepts from transformatics in genetics, in a clearer, straight forward manner.

For starters, we can certainly say that irrespective of how long or from where a particular DNA sequence, Θ_{DNA} originated from, once any sequence of data (e.g unstructured or unprocessed raw genetic code sequence dump), a string (say a proprietary encoding of some DNA code), a name (say of a particular species of interest and whose actual/representative genome sequence is know) or even a number (an id of a genome sequence in some standard genome database, etc.) is mapped to standard DNA sequence code, we shall know that any such transformation or mapping obeys **Theorem 1**:

Theorem 1 (ψ_{DNA} is the alphabet of any DNA sequence)**.** *For any sequence of DNA, Θ_{DNA}, produced from some or any other source Θ, irrespective of whether that source is itself a DNA sequence or not, we know that such an encoding or mapping, if it results in a valid DNA sequence Θ_{DNA}, obeys the following general transformation:*

Transformation 1. $\Theta \rightarrow \Theta_{DNA}$;

$$\forall a_{i\in[1,\mathcal{U}(\Theta_{DNA})]} \in \Theta_{DNA} \quad \exists \rho \in \psi_{DNA} : a_i = \rho$$

Proof. Assume Θ_{DNA} contains some symbol α such that $\alpha \notin \psi_{DNA}$, it would contradict **Definition 1** which clearly states the legitimate membership of any DNA sequence. □

And definitely, the equivalent truth for RNA sequences would likewise follow from **Definition 4**. So, we can then correctly tell that a sequence such as $\Theta_{insulin}$, which is the genetic code sequence defined as:

$$\begin{aligned}\Theta_{insulin} = {} & \texttt{ATGGCCCTGTGGATGCGCCTCCTGCCCCTGCTGGCGCTGCTGGCCCTCTGGGGACC} \\ & \texttt{CCAGCCGCAGCCTTTGTGAACCAACACCTGTGCGGCTCACACCTGGTGGAAGCTCTCTAC} \\ & \texttt{CTAGTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGGAGGCAGAGGAC} \\ & \texttt{CTGCAGGTGGGGCAGGTGGAGCTGGGCGGGGGCCCTGGTGCAGGCAGCCTGCAGCCCTTG} \\ & \texttt{GCCCTGGAGGGGTCCCTGCAGAAGCGTGGCATTGTGGAACAATGCTGTACCAGCATCTGC} \\ & \texttt{TCCCTCTACCAGCTGGAGAACTACTGCAACTAG} \end{aligned} \tag{6}$$

And which sequence[1], despite having been obtained from an authority — the **Leiden Open Variation Database (LOVD)**, which is based on NCBI's RefSeq data[20], might need to be verified by hand or with a critical eye, and that's where a law such as **Theorem 1** would come in handy. Also, note that, unlike common sequence notations we might see in this work or that we have encountered before, we wrote $\Theta_{insulin}$ in a manner somewhat more convenient for the kind of verbose sequence that DNA code generally is. The notation — without opening or closing brackets around the sequence terms and neither commas in them might be reminiscent of the *String [Chart] Sequence* notation we developed recently (see **Transformation 15** in [6]).

We might for example start to wonder, how might we go about determining if some two DNA sequences, Θ_1 and Θ_2 are the same or perhaps if they share parts of each other, and by how much? We might wonder if they are the same sequence merely shuffled/anagrammatized — since we saw that this can occur during natural processes such as meiosis[10]. In case they are different, we might want to for example quantify the relative frequency of their distinct members, e.g mapping ψ_{DNA} or ψ_{RNA} to a sequence of relative frequencies, etc. These are all things we shall soon look into and know very well how to do or talk about, using genetics terms and the mathematical language and techniques from transformatics.

[1] $\Theta_{insulin}$ is the short coding DNA sequence (CDS) for the human **INS** gene — the part that gets transcribed into mRNA and translated by ribosomes into the insulin protein[14]

3 Sequence Abstraction using Symbol Sets: From Flat-Structure Sequences to Higher-Level Sequences

Especially for nucleic acid sequences, but also for any other kind of sequence Θ, there might be legitimate situations in which looking at or dealing with the flat-structure sequence — such as for the actually modest[2] $\Theta_{insulin}$, might be cumbersome. And so, we are going to briefly look at ways that we might re-write verbose flat-sequences in more terse or higher-level abstract ways so as to focus on details that the flat-structure perhaps doesn't clearly express. This for example is naturally relevant in nucleic acid sequences since they are for example useful in scenarios where their processing is done in n-tuples/n-grams — such as with codons (nucleotide 3-grams) or as gene programs (self-contained sequences of codons), etc.

So, assuming we have some sequence of symbols Θ_1 defined as such:

$$\Theta_1 = \langle a_1, a_2, ..., a_n \rangle \tag{7}$$

If we wish to re-write the same sequence such that every k terms are grouped in the same subsequence, we can then re-write Θ_1 differently as such:

$$\Theta_2 = \Theta_2(n,k) = \langle a_{11}, a_{22}, a_{33}, a_{4k}, a_{51}, a_{62}, ..., a_{(n-k+1)(1)}, ..., a_{(n-k+k-1)(k-1)}, a_{(n-k+k)(k)} \rangle \tag{8}$$

Of course, in **Equation 8** we have somewhat wanted to extrapolate well and illustrate what would happen in an informative case such as when $k = 4$ and n is not only significantly larger than k, but is also its perfect multiple. Otherwise, we might more concisely write that same sequence as:

$$\Theta_2(n,k) = \langle \langle a_1, a_2, a_3, a_4 \rangle_1, \langle a_5, a_6, ..., a_8 \rangle_2, ..., \langle a_{(n-k+1)}, ..., a_{(n-k+k-1)}, a_{(n-k+k)} \rangle_{\frac{n}{k}} \rangle \tag{9}$$

So, this expression in **Equation 9** is our first proper appreciation of what higher-level sequences might be like when they are an abstraction of normal flat-structure sequences. We see for example here, that the $k = 4$ **bundling-parameter** means, we shall process the original flat sequence k items at a time, and so, each subsequence in $\Theta_2(n, 4)$ then contains exactly (or at most) 4 items — we are assuming 4 is a factor of n to keep things simple. And so, we expect that, in total, we should have $\frac{n}{k}$ subsequences after applying the bundling transform:

Transformer 1 (The **k-GRAM Generator**). $\Theta = \langle a_1, a_2, ..., a_n \rangle \xrightarrow{O_{bundle-k}(\cdot)} \Theta^*$;

[2] 'Modest' because, genomes and realistic genome sequences can typically span millions of genes or billions of nucleotides. We for example know that for a typical human cell's DNA there are ≈3.2 billion base pairs[21], and yet we know each base pair consists of two nucleotides, one for each strand of the double-helix (so that's ≈6.4 billion), plus a few more (≈16,569 base pairs) from mitochondrial DNA[22]

$\Theta^* = \langle\langle a_1, a_2, ..., a_k\rangle_1, \langle a_{k+1}, ..., a_{2k}\rangle_2, ..., \langle a_{(n-k+1)}, ..., a_{(n-k+k-1)}, a_{(n-k+k)}\rangle_{\frac{n}{k}}\rangle$
$\forall a_i \in \Theta \quad \exists a_{ij} \in \Theta^* : a_i = a_{ij} \quad \wedge \quad j \in [1, k]$
For some $n, k \in \mathbb{N}$*, and that* $\text{ɯ}(\Theta^*) = \frac{n}{k}$ *= number of k-gram subsequences generated from* Θ*.* $\square$

Thus, we see that the sequence depicted in **Equation 9** with $k = 4$ could as well be produced by the generator defined in **Transformer 1**. But not just that, if we applied that transformer to the DNA sequence $\Theta_{insulin}$ first presented in **Section 2** — see **Equation 6**, we can then produce a proper standard codon/3-gram mapping of that DNA sequence that would look something like:

$$\Theta_{insulin} \xrightarrow{O_{bundle-3}(\cdot)} \langle \texttt{<ATG>,<GCC>,<CTG>,<TGG>,<ATG>,<CGC>,<CTC>,...,<ACC>} \\ \texttt{...,<TAC>,<TGC>,<AAC>,<TAG>}\rangle \quad (10)$$

And the way we have bundled up the nucleotides in **Equation 10** is exactly how a natural sequence processor such as a Ribosome (see details in **Definition 10**) would process it so as to produce say a corresponding protein.

Of course, though we might not delve into it here or at the moment, we know that by chaining several sequence transformers — such that one operates on the output of the previous one to produce the next sequence (see **Transformer 17** as an example), we can arrive at even higher abstraction sequences that might render certain sequence processing programs easier to write, analyze or define. For example, we know that, much as a Ribosome Processor operates on a sequence of codons, and yet, any legitimate protein synthesis and genetic code sequence processing program will require some sort of necessary delimiters (such as the START and STOP codons — refer to the Protein Synthesis Flow Chart in **Figure 6**), and so that, we might (if we could), abstract away codons and instead generate from a flat-structure DNA or perhaps mRNA sequence, a higher-level n-gram sequence of genes or **gene-program sequences**.

4 Sequence Analysis Using Symbol Sets

We now turn our attention to the matter of using ideas from transformatics to help analyze or understand genetic code sequences. For starters, we shall want to consider the matter of how to tell how far apart or dissimilar any two nucleic acid sequences might be. Of course, consequently, that would also translate into meaningful way to tell how far apart not just low-level genetic-creations such as tissue, cell-types, proteins, etc might be based on the genetic code that produces them, but also how different entire organisms (within the same species or not) might be.

Because we can learn much about an organism based on its characteristic

genome, and since that can be mapped to a flat-structure sequence of symbols such as we have already encountered in earlier sections, it might start to make sense to leverage the similarity/distance measures we have already developed so as to apply them to quantifying the distance between say species. In an earlier paper[23], we have introduced and also demonstrated how a sequence-analysis measure called the **Anagram Distance Measure**, $\tilde{A}(\Theta, \Theta^*)$, might be used to quantify how far apart some two sequences Θ and Θ^*, that for the simplest scenarios better be of the same length, can be compared based on their common membership and the relative lexical ordering of the members in either sequence.

In the simple analysis we shall use to illustrate or develop our concepts, we shall deal with hypothetical nucleic acid sequences mostly — especially DNA sequences, and their lengths shall be kept short to keep our analyzes simple, but also, their membership — in terms of nucleotides, might have nothing to do with actual/natural sequences. The concepts and ideas we shall develop though, should be readily applicable to any or most nucleic acid sequences if not any sequences in general.

First, assume we have the following three sequences:

1. $\Theta_1 = CATGGGACTGCC$
2. $\Theta_2 = ATAATAAGAGGGATCTGA$
3. $\Theta_3 = AUAGGGAGAAUC$

We can start by attempting to answer the question: **Which of these sequences are DNA and which are RNA?**

So, to answer that question, we can start by reducing each of those sequences to their respective [ordered] sequence symbol sets. In fact, if we first relax the assumption that they are either DNA or RNA sequences, and merely look at them as though they were any kind of sequence (for which we don't know the base or base-symbol set), then we can merely use the definition of an **Unspecific Symbol Set of Θ in Any Base** — see **Definition 4** in [18]. The algorithm for how to do that is simple and is well presented in that definition — we basically create another sequence, in which we insert each distinct symbol from Θ that we encounter while processing the sequence from Left-to-Right[3]. Thus, we might construct the relevant transformer for this as such:

Transformer 2 (**Sequence Unspecified Symbol Set Generator**).

$$\Theta \xrightarrow{O_{suss-gen}(\cdot)} \Theta^* \quad ; \quad \mathscr{u}(\Theta^*) \leq \mathscr{u}(\Theta) \quad \wedge \quad \Theta^* = \hat{\psi}(\Theta)$$

And thus, applying that transformer to the three sequences we have, we shall obtain the following results:

[3]Such that the leftmost term is the first to be processed.

Transformation 2. $\Theta_1 \xrightarrow{O_{suss-gen}(\cdot)} \langle C, A, T, G \rangle$

Transformation 3. $\Theta_2 \xrightarrow{O_{suss-gen}(\cdot)} \langle A, T, G, C \rangle$

Transformation 4. $\Theta_3 \xrightarrow{O_{suss-gen}(\cdot)} \langle A, U, G, C \rangle$

At this juncture, we can then closely inspect the resultant sequences — each of them a kind of symbol set, and then judge which of ψ_{DNA} or ψ_{RNA} they are associated with. To proceed in a rigorous manner, we might also want to compute the **Natural Symbol Set of Θ in some Base-β** — see **Definition 5** in [18]. A suitable transformer to compute such a symbol-set (for which, unlike the unspecific symbol set, orders the terms in the resultant sequence by their natural order of occurrence in the base's ordered symbol set) would as as such:

Transformer 3 (**Sequence β-Natural Symbol Set Generator**).
$\Theta \xrightarrow{O_{snss-gen-\beta}(\cdot)} \Theta^* \quad ; \quad \mathcal{U}(\Theta^*) \leq \mathcal{U}(\Theta) \quad \wedge \quad \Theta^* = \psi_\beta(\Theta)$

And, since we already have an idea what the symbol sets for each of the three sequences are — from which we can guess and/or disqualify some candidate bases — e.g, since $\hat{\psi}(\Theta_3) \setminus \psi_{DNA} = \{U\}$, then Θ_3 can't be a DNA sequence. However, to complete our analysis, note that:

Transformation 5. $\Theta_1 \xrightarrow{O_{snss-gen-DNA}(\cdot)} \langle A, C, G, T \rangle$

Transformation 6. $\Theta_2 \xrightarrow{O_{suss-gen}(\cdot)} \langle A, C, G, T \rangle$

Transformation 7. $\Theta_3 \xrightarrow{O_{suss-gen}(\cdot)} \langle A, C, G, U \rangle$

So, we can safely conclude that:

1. Since $\psi_{DNA}(\Theta_1) = \psi_{DNA}$, then Θ_1 is a DNA sequence.
2. Since $\psi_{DNA}(\Theta_2) = \psi_{DNA}$, then Θ_2 is a DNA sequence.
3. Since $\psi_{RNA}(\Theta_3) = \psi_{RNA}$, then Θ_3 is a RNA sequence.
4. Even if we forced it, note that $\psi_{RNA}(\Theta_1) = \langle A, C, G, T \rangle \neq \psi_{RNA}$, or rather that $\hat{\psi}(\Theta_1) \setminus \psi_{RNA} = \{T\}$ so Θ_1 can't be an RNA sequence.

Talking of which, in case we had some variation of Θ_1 that has no instances of T in it, such as the sequence $\Theta_4 = CAAGGGACAGCC$, then we shall find that:

Transformation 8. $\Theta_4 \xrightarrow{O_{suss-gen}(\cdot)} \langle C, A, G \rangle$

and that:

Transformation 9. $\Theta_4 \xrightarrow{O_{snss-gen-DNA}(\cdot)} \langle A, C, G \rangle \subset \psi_{DNA}$

But also that:

Transformation 10. $\Theta_4 \xrightarrow{O_{snss-gen-RNA}(\cdot)} \langle A, C, G\rangle \subset \psi_{RNA}$

And thus, we have the peculiar case in which a nucleic acid sequence can both be a legitimate DNA or RNA sequence! However, such a truly peculiar sequence it is! Because, given the START-codon symbol set, $\psi_{na-START}$, a sequence of all the "start" kind codons whether of DNA or RNA type, is defined as such from all we currently know:

Definition 5 (The **Nucleic Acid START-codons**)**.** *For any known organism, prokaryote or eukaryote, the only legitimate and known codons of the START-kind are any of the codons or START 3-grams in the unordered sequence $\psi_{na-START}$, the **nucleic acid START-codon symbol set**.*

$$\psi_{na-START} = \{ATG, AUG, GTG, GUG, TTG, UUG\} \tag{11}$$

*So, that, if say a nucleic acid program for producing some protein via some gene program Θ_{na} were to be processed by an ideal ribosome (see **Definition 10**), the ribosome would never produce anything — or, equivalently, there is no guarantee that any protein would be produced no matter what or any instructions the gene program contains, as long as it doesn't contain any one of the members of $\psi_{na-START}$.*

One interesting consequence of that definition is the following law:

Law 2 (**Non-Coding Gene Programs**[4])**.** *If any gene-program Θ_{na} is processed by a ribosome, and yet it has the property:*

$$\psi_{DNA}(\Theta_n a) \cap \psi_{na-START} = \emptyset \quad \vee \quad \psi_{RNA}(\Theta_n a) \cap \psi_{na-START} = \emptyset$$

It implies that Θ_{na} shall never be transcribed by the ribosome, and neither shall it ever result in any new protein or new amino-acid product within the containing cell system[5]*.*

So, that, if say a nucleic acid program for producing some protein via some gene program Θ_{na} were to be processed by an ideal ribosome processor (see **Definition 10**), the ribosome would never return — or, equivalently, there is no guarantee that any products — neither amino-acid, nor completed/new protein, shall be synthesized by the ribosome.

[4]Also known as **Introns** in some contexts.
[5]Refer to ribosome definition for details: **Definition 10**

Of course, it's very likely that such gene sequences exist, and there might be nothing wrong with them being natural too. However, we shall want to learn more about this from the domain experts in the future.

That said, another, closely related case is that of the consequences of the STOP-codons or rather, the STOP-codon symbol set, $\psi_{na-STOP}$ defined as such:

Definition 6 (The **Nucleic Acid STOP-codons**)**.** *For any known organism, prokaryote or eukaryote, the only legitimate and known codons of the STOP-kind are any of the codons or STOP 3-grams in the unordered sequence $\psi_{na-STOP}$, the* ***nucleic acid STOP-codon symbol set****.*

$$\psi_{na-STOP} = \{TAA, UAA, TAG, UAG, TGA, UGA\} \tag{12}$$

Among important consequences of **Definition 6**, is that, if say a nucleic acid program for producing some protein via some gene program Θ_{na} were to ever be processed by an ideal ribosome (see **Definition 10**), the ribosome might produce something — some amino-acids for example, but would never return — equivalently, there is no guarantee that any protein would be released by the protein manufacturing process/ribosome, no matter what or how many any amino-acid productions and translate instructions the gene program contains and which have been executed by the ribosome. As long as the gene-program doesn't contain any one of the members of $\psi_{na-STOP}$ that is.

It is not immediately clear what the plausibility of such an *evil gene-program* existing out there in nature might be, but given normal genes sometimes mutate[2], it can't be ruled out that such awkward programs might exist. Though we won't dive into that here, who knows... from a computer security perspective, such a program might cause system errors or faults such as a System-Out-Of-Resources problem given the ribosome might attempt to produce an infinite length amino-acid, or perhaps that the cell's protein manufacturing closure might run out of space, or that the processor might end up hanging since the factory never is able to reach any of WAIT, DETACH or RESET states — see **The Ribosome State Machine** in **Figure 7** but also refer to the Protein Synthesis Process in **Figure 6** to clearly, logically appreciate these kinds of problems and/or corner-case scenarios.

That said, in case a normally correct gene-program such as what we saw in $\psi_{insulin}$ (see **Equation 6**) is altered with — perhaps by a natural mutation, or perhaps a virus, or even in a totally malicious feat of gene-manipulation medicine, and such a transformation results in a nucleic-acid sequence satisfying any of the above queer cases, who knows, but it might as well be the root case of some

difficult flaw in the organism — an incurable and fatal disease (if the faulty gene-program is crucial for normal life support and that it has no alternatives or simple way to be solved), and these might the kinds of hard problems that might require not just medical doctors, biologists or perhaps genetists to tackle, but also people like computer hackers, information-security experts and software debuggers[6]!

5 Sequence Analysis Using Complement Sets

The idea of complements when applied to na-Sequences has already been introduced in **Section 1**, and we get to see it depicted in an exemplary typical diagrammatic depiction of the DNA structure in **Figure 1**. However, apart from having talked about the fact that nucleic acids typically or naturally occur together in base-pairs depicting pairwise complements, we might want to leverage transformatics to study this idea at a more general level — say, to explore the matter of **na-Sequence** (as well as *any sequence*) complements, and thus this section.

Assume we start by exploring simple binary sequences — especially because bits help generalize many ideas in theory and practice. We can for example start by noticing that, like in the case of the interesting **Lu-Number System**[25] — in which any kind of *basic information* can be abstracted by mapping it to either signal (¿) or anti-signal (¡), and that *complex information* can then be generated from that using transforms on basic LNS sequences or expressions, etc. We can start by looking at what can be said of the transforms on the binary sequence symbol set, ψ_{bin} — or better, ψ_{01}. In our formulations, we shall denote the complement of a symbol ρ by $\neg\rho$, and for a sequence Θ, with $\neg\Theta$.

[6]It shouldn't come as a surprise, that with a stronger marriage of computing theory and biology, or medicine, that certain problems in nature — not just with humans or animals, but also with plants for example, and especially if they are either inheritable — meaning genetic, or if they might somehow be remedied via clever hacking or debugging and modifying of the organism via gene programs, might call for the skills of and expertise of software engineers and program debuggers from hardcore computer science and not just the medical or biological sciences! A great background work on this subject by the author might come in handy — refer to [24]

Sequence Name	Sequence	Formula	Note
$\Theta 1$	0 1	ψ_{01}	Binary Symbol Set as a basic Sequence
$\Theta 2$	0 1 0 1	$\psi_{01} \cdot \psi_{01}$	Sequence Concatenation/Multiplication
$\Theta 3$	0 2 0 1	$\psi_{01}(1 + \psi_{01})$	Sequence as Linear Combination of other Sequences: case of memberwise addition of the above two sequences
$\Theta 4$	1 0 1 0	$\neg\Theta 2 = \neg\psi_{01} \cdot \neg\psi_{01}$	Basic Complement Transforms

Table 1: An example of exploring symbol and sequence complements in sequence transformations

So, we can begin by noting that, if some sequence Θ spans some symbol set ψ_β — such as how all the sequences in **Table 1** span ψ_{01}, then we can say that the complement sequence, $\neg\Theta$ spans the complement symbol set $\neg\psi_\beta$. But not just that. You can readily tell from studying that table, that actually, if sequence Θ spans ψ_β, then it also spans $\neg\psi_\beta$ — meaning, its members essentially are one of the distinct elements from either symbol set even though the two sets might not necessarily have the same ordering of their members.

Further, note that, if ψ_β is the symbol set of some sequence Θ, and that the **complement symbol set** it is equivalent to is $\neg\psi_\beta$, then we can comfortably say that $\neg\Theta$ spans $\neg\psi_\beta$ as well. This is true, because, for any sequence Θ, its complement, $\neg\Theta$ is the sequence of the memberwise complements of Θ. That is to say:

$$\psi_\beta = \neg(\neg\psi_\beta) \tag{13}$$

and

$$\Theta = \neg(\neg\Theta) \implies \neg\Theta = \neg(\Theta) \tag{14}$$

So that, if

$$\psi_\beta = \langle \prod_{i=1}^{n} a_i \rangle \quad \implies \quad \neg\psi_\beta = \langle \prod_{i=1}^{n} a_{n-i} \rangle \tag{15}$$

And for any $\Theta : \mathbb{N} \times \psi_\beta$ we know that:

$$\neg\Theta = \prod_{i=1}^{n} \neg(a_i) \tag{16}$$

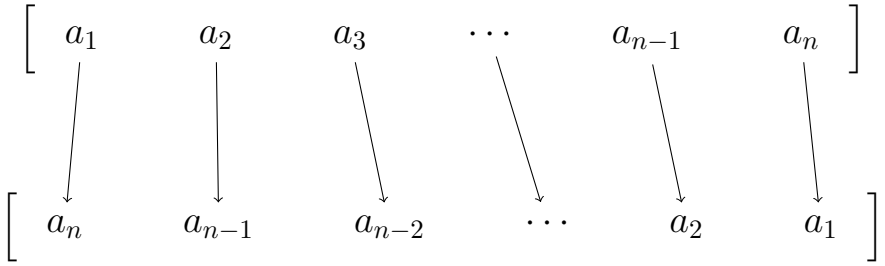
for $n = \mathcal{U}(\Theta)$ and that $\forall a_i \in \psi_\beta$,

$$\neg(a_i) = \rho_i \in \neg\psi_\beta \tag{17}$$

So, one way to appreciate **Equation 17**, is by realizing that by knowing the members and their order in ψ_β, we can then systematically determine the members and ordering of $\neg\psi_\beta$ via **Equation 15**, and so that, for some a_i in ψ_β, its complement in $\neg\psi_\beta$ is the term at position index $n-1$ in ψ_β. Thus, as we saw in **Table 1**, for $a_0 = 0$ in ψ_{01}, the complement, $\neg a_0 = a_{\mathcal{U}(\psi_{01})-0} = a_2 = 1$. This mechanics can be extended or applied to sequences of arbitrary length and composition as necessary. For purposes of future use, we shall generalize this with a useful theorem as such:

Theorem 2 (**Complement Sequences**)**.** *For any ordered sequence* $\Theta_n\langle\prod_{i=1}^{n} a_i\rangle$*, its equivalent* ***complement sequence****, also denoted* $\neg\Theta_n$*, is the sequence with the signature* $\Omega_n\langle\prod_{i=1}^{n} \rho_i\rangle$ *such that* $\rho_i = a_{n-1} \quad \forall a_i \in \Theta_n$*. We shall also write* Ω_n *as* $\neg\Theta_n$ *where necessary.*

Proof. By **Equation 15**, we see that if $\Theta_n = \langle a_1, a_2, a_3, ..., a_{n-1}, a_n\rangle$, then we can derive its complement $\neg\Theta_n$ via the mapping of its members to corresponding members in its lateral inverse as such:



□

And since we are dealing with sequence transformations in this work, we might as well define the necessary transformer:

Transformer 4 (The **Complement Transformer**)**.**
$\Theta \xrightarrow{O_{complement}(\Theta)} \Theta^*; \quad \Theta^* = \neg\Theta$

5.1 Significance of Sequence Complements, $\neg\Theta$ and Complement Symbol Sets, $\neg\psi_\beta$

First, we shall momentarily return to binary sequences.

We note that, for some binary sequence $\Theta : \mathbb{N} \times \psi_{01}$, the transform

Transformation 11. $\Theta \xrightarrow{O_{complement}(\cdot)} \Theta^*; \Theta^* = \neg\Theta$

which applies **Transformer 4**, produces the same sequence as Θ but with all bits swapped by their complements, which, as per usual nomenclature in computer science for example, means the resultant sequence is produced from the source via a **bitwise NOT operation**, and which, for binary sequences, is as simple as merely flipping the bits individually.

Thus, if our source sequence was

$$\Theta = \langle 01011110 \rangle \tag{18}$$

Then

Transformation 12. $\Theta \xrightarrow{O_{complement}(\Theta)} \langle 10100001 \rangle$

And then, returning to nucleic acid sequences, we note that if we for example have some na-Sequence such as

$$\Theta = \langle ATCGCGTAT \rangle \tag{19}$$

That we wish to treat as a DNA-sequence, then its *na-Complement* sequence would be derivable from ψ_{DNA} as such:

Since $\psi_{DNA} = \langle A, C, G, T \rangle$, then by **Equation 15**, we know that:

$$\neg(\psi_{DNA}) = \neg\psi_{DNA}) = \langle T, G, C, A \rangle \tag{20}$$

So that, $\forall\rho \in \psi_{DNA}$, the equivalent complement is via the mapping:

$$\begin{bmatrix} A & C & G & T \end{bmatrix}$$
$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$
$$\begin{bmatrix} T & G & C & A \end{bmatrix}$$

Which is also the consequence of **Theorem 2**. Thus, for the sequence in **Equation 19**, the corresponding complement DNA sequence is:

$$\neg\Theta = \langle TAGCGCATA \rangle \tag{21}$$

This ideal can be further appreciated by noticing that the traditional DNA sequence base symbol set (which we shall denote by $\dot{\psi}_{DNA}$), is actually different from ψ_{DNA}, and is defined as:

$$\dot{\psi}_{DNA} = \langle A, T, C, G \rangle \tag{22}$$

We should come to appreciate this special symbol set, or even justify it — for logical and mathematical analyses, by acknowledging that it depicts not just a sequence of distinct symbols from ψ_{DNA}, but also their natural order based on both order of occurrence in ψ_{DNA} **and** their equivalent **symbol complements** in $\neg\psi_{DNA}$! It is such a terrific discovery/realization. And before we proceed, we note that the $\dot{\psi}_{DNA} \rightarrow \neg\dot{\psi}_{DNA}$ mapping for pairwise-ordered DNA sequences is as such:

$$\begin{bmatrix} A & T & C & G \end{bmatrix} \\ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ \begin{bmatrix} G & C & T & A \end{bmatrix}$$

Because, from **Equation 22**, we can tell that:

$$\neg\dot{\psi}_{DNA} = \langle G, C, T, A \rangle \tag{23}$$

Perhaps, and worth checking out, even if momentarily, what would happen when the idea of complement symbols sets and complement sequences is applied to the more commonplace base-10 sequences (aka "usual numbers")?

First, note that since ψ_{10} — our choice of symbol for the decimal symbol set, is expressed as such:

$$\psi_{10} = \langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \rangle \tag{24}$$

Then, by **Theorem 2**, its complement is:

$$\neg\psi_{10} = \langle 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 \rangle \tag{25}$$

So that, for any base-10 digit, the corresponding **complement digit** is as in the mapping below:

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix} \\ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ \begin{bmatrix} 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}$$

Such a peculiar cipher! Because, usual/familiar number sequences such as the **Hi-Fi o-SSI**[23], defined as such:

$$\Theta_{HiFi} = 8649137520 \tag{26}$$

Has the complement:

$$\neg\Theta_{HiFi} = 1350862479 \tag{27}$$

Which, given how that special sequence was derived/discovered(see **Section 5.1** in [18] or **Equation 19** in the same paper), it is interesting to note that its complement under ψ_{10} is also an *orthogonal symbol set identity*! For those who have studied the o-SSI paper[18], this can really be exciting for the study of symbol sets and sequences in general — yes, even for nucleic acid sequences! But, that said, of course, it shouldn't come as a surprise, since, in simple terms, a sequence's complement is just one very special kind anagram among the set of all potential anagrams it can have, and so, before we leave the base-10 case, we can note that if some sequence $\Theta_{10} : \mathbb{N} \times \psi_{10}$ is a base-10 o-SSI, then its complement is also a base-10 o-SSI, and in fact, in general — even for non-numeric symbol sets such as ψ_{na}, ψ_{DNA} or special ψ_{RNA}, that:

Theorem 3 (**Complement of an o-SSI is an o-SSI**). *If some sequence* $\Theta : \mathbb{N} \times \psi_\beta$ *is an orthogonal symbol set identity for the base* β*, then its complement,* $\neg\Theta$ *is also an orthogonal symbol set identity for* β*.*

5.2 Potential Applications of Sequence Complements or Inverses

First, in genetics and analysis of genetic code sequences, note that for especially sequences in say the *non-native*(non-storage) form, such as the processed/derived RNA-sequences, it might sometimes be useful to determine if some two sequences are the same, equivalent or closely related, even though their apparent ordering of members, or even member composition looks unlike, **as long as their cardinalities are the same**[7].

We should note, and interestingly so, that the complement of any sequence Θ, which is $\neg\Theta$, is just an anagram of Θ, and that in general, we can comfortably say:

Theorem 4 (**Equivalence of Sequences under the Complement Transform**). *If two sequences* Θ *and* Ω *that are* ***not equal****, but which do have the same cardinality and similar distribution of members, then they are essentially equivalent under the sequence complement transform.*

Proof. The proofs are several:

1. Since $\text{ɯ}(\Theta) = \text{ɯ}(\Omega)$ and $\forall\alpha \in \Theta \;\; \exists\rho \in \Omega : \alpha = \rho$ or equivalently, that $\forall a_i \in \Theta, \text{ɯ}(a_i \in \Theta) = \text{ɯ}(a_i \in \Omega)$, then there exists some anagram of Θ, denoted Θ^* such that $\Theta^* \xrightarrow{O_{complement}(\cdot)} \Omega$, which is possible by **Theorem 2**.

[7]This requirement to have their sizes the same, is a natural consequence of how a sequence and its anagram relate — see [23]

2. Because sequence complements are anagrams of each other, then if the two sequences are complements of each other, we expect that by the definition of the complement transformer (see **Transformer 4**), $\Omega = \neg\Theta$ or that $\Theta = \neg\Omega$.
3. We might also proceed by use of **modal sequences**(see **Definition 1** in [6]): Given the conditions on the two sequences, we then expect that if Θ is a sequence complement of Ω or vice-versa, then, computing their corresponding modal sequence statistic should result in the following result: $\overset{>}{\Theta} = \neg(\overset{>}{\neg\Omega})$ or equivalently, $\overset{>}{\Omega} = \neg(\overset{>}{\neg\Theta})$

□

Note that, for RNA, since $\psi_{RNA} = \langle A, C, G, U\rangle$ (**Equation 5**), then:

$$\neg\psi_{RNA} = \langle U, G, C, A\rangle \tag{28}$$

And this is what we'd expect via the complement mapping:

$$\begin{bmatrix} A & C & G & U \end{bmatrix}$$
$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$
$$\begin{bmatrix} U & G & C & A \end{bmatrix}$$

But, since the traditional pairwise ordered symbol set for RNA, $\dot{\psi}_{RNA}$ is

$$\dot{\psi}_{RNA} = \langle A, U, C, G\rangle \tag{29}$$

Then its corresponding complement,$\neg\dot{\psi}_{RNA}$, is

$$\neg\dot{\psi}_{RNA} = \langle G, C, U, A\rangle \tag{30}$$

And so we have the associated complement mapping as

$$\begin{bmatrix} A & U & C & G \end{bmatrix}$$
$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$
$$\begin{bmatrix} G & C & U & A \end{bmatrix}$$

So, that, for some RNA-sequence such as Θ_{met}:

$$\Theta_{met} = \langle A, U, G\rangle \tag{31}$$

Which is the famous START-codon **Methionine** (see **Table 5**), has as its corresponding nucleic acid sequence complement under $\dot{\psi}_{RNA}$:

Transformation 13. $\Theta_{met} \xrightarrow{O_{complement}(\Theta,\dot{\psi}_{RNA})} \langle G, C, A\rangle$
*That codon isn't the same as the source (ATG/AUG/Methionine), but there's some debate concerning what it actually is called — **Table 1** in [2] names codon GCA as **Arginine**, while Wikipedia[26] names it **Alanine**.*

While — **and important to note**, its complement under ψ_{RNA} is [indeed] different!

Transformation 14. $\Theta_{met} \xrightarrow{O_{complement}(\Theta,\psi_{RNA})} \langle U, A, C\rangle$
and we know that codon UAC is Tyrosine[26][8].

It should perhaps be immediately noticeable that the ability to derive new/different/special sequences such as complement codons from ordinary na-Sequences might have some useful if not exciting applications as well as problems it awakens for the geneticists, molecular biologists and researchers of gene-sequencing[9] and analysis. For example, apart from acknowledging that na-Sequence's complements denote potentially legitimate na-helix strand partners for any two sequences (in the form Θ and $\neg\Theta$), it would be interesting to determine what other important things might we learn, discover from exploring these kinds of sequence transforms?

It's already interesting to note that under complement transforms (which are special kinds of anagrams), the natural/universal protein synthesis START-codon AUG/ATG is equivalent to GCA (**Transformation 13**) or UAC (**Transformation 14**), but, **are those other codons also START-codons?** That said, is there a possibility that some natural process or genetic code translation or transformation might naturally produce ATG or AUG from any of those corresponding [symbolic] complement sequences? What would this teach us about natural/biological processors? What of the possibilities of using synthetic/artificial processors to render such complement operations useful?

6 Sequence Analysis Using The Anagram Distance and the Modal Sequence Statistic

We are going to talk about the matter of comparing or analyzing two or more na-Sequences using especially statistical measures that have been set down by transformatics theory for especially ordered sequences.

[8] Concerning this particular codon, TAC/UAC, we find that though the Wikipedia reference tables assign it the name Tyrosine in both the DNA and RNA encodings, and yet, the physical reference we have at hand[2] (perhaps now out-of-date), assigns the name "Methionine" to TAC — Table 1 in that reference book didn't list RNA codes, but we still would expect to use TAC to look-up the correct name for UAC.

[9] For those that don't know: Gene sequencing is the process of determining the exact order of nucleotides — A, T, C and G — in a segment of DNA that makes up a gene. It's like reading the biological "code" that instructs cells how to build proteins and regulate functions[14].

6.1 Six Sequences and Four Scenarios

In particular, we shall be looking at na-Sequence analysis through lens of measures we have formalized earlier on, and shall break down our analysis into four representative scenarios:

1. **Scenario A:** Given some two particular instances of na-Sequences, $\Theta 1_n$ and $\Theta 2_n$, of the same length, n, and with similar symbol sets ($\psi_{na}(\Theta 1_n) = \psi_{na}(\Theta 2_n)$), how to quantify how similar or dissimilar they are?
2. **Scenario B:** Given some two particular instances of na-Sequences, $\Theta 1_n$ and $\Theta 2_k$, of different lengths, n and k, but with similar symbol sets ($\psi_{na}(\Theta 1_n) = \psi_{na}(\Theta 2_k)$), how to quantify how similar or dissimilar they are?
3. **Scenario C:** Given some two particular instances of na-Sequences, $\Theta 1_n$ and $\Theta 2_k$, of different lengths, n and k and with different symbol sets ($\psi_{na}(\Theta 1_n) \neq \psi_{na}(\Theta 2_k)$), how to quantify how similar or dissimilar they are?
4. **Scenario D:** Finally, how, when given a particular na-Sequence, Q_k, and two collections/samples of other sequences $(\Theta 1, \Theta 2, \Theta 3, ..., \Theta m)$ and $(\Omega 1, \Omega 2, \Omega 3, ..., \Omega n)$ of potentially dissimilar lengths, and that belong to labeled or classified populations — POPA and POPB, to determine which of the two populations the query sequence, Q_k best-belongs to (so we classify it under one of the labels), as well as determine which of the sample member sequences from either collections it is closest to?

Note that in conducting these analyses, we shall especially exploit the sequence-specific measures, $\tilde{A}(\cdot)$, the Anagram Distance Measure[23][6], and $\overset{>}{\Theta}$, the Modal Sequence Statistic[6], and that we shall use the following representative, even though somewhat hypothetical (for the first four), na-Sequences:

$$\Theta 1_{DNA} = \langle TACGGGCATTCC \rangle \tag{32}$$

$$\Theta 2_{RNA} = \langle AUGAUGCGCGGGCATAUC \rangle \tag{33}$$

$$\Theta 3_{DNA} = \langle ATAATGGGGGCATTGA \rangle \tag{34}$$

$$\Theta 4_{DNA} = \langle TACTCCGGGCAT \rangle \tag{35}$$

$$\Theta_{insulin} = \texttt{ATGGCCCTGTGGATGCGCCTCCTGCCCCTGCTGGCGCTGCTGGCCCTCTGGGGACC}$$
$$\texttt{CCAGCCGCAGCCTTTGTGAACCAACACCTGTGCGGCTCACACCTGGTGGAAGCTCTCTAC}$$
$$\texttt{CTAGTGTGCGGGGAACGAGGCTTCTTCTACACACCCAAGACCCGCCGGGAGGCAGAGGAC}$$
$$\texttt{CTGCAGGTGGGGCAGGTGGAGCTGGGCGGGGGCCCTGGTGCAGGCAGCCTGCAGCCCTTG}$$
$$\texttt{GCCCTGGAGGGGTCCCTGCAGAAGCGTGGCATTGTGGAACAATGCTGTACCAGCATCTGC}$$
$$\texttt{TCCCTCTACCAGCTGGAGAACTACTGCAACTAG} \quad (36)$$

$$
\begin{aligned}
\Theta_{rbcl} = {} & AUGAAUUGCUUUGGAGGAGUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAG\\
& UUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAG\\
& GAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUG\\
& GAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCU\\
& UUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUG\\
& GCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCU\\
& UUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUU\\
& GCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAG\\
& UUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAG\\
& GAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUG\\
& GAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCU\\
& UUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUG\\
& GCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCU\\
& UUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUU\\
& GCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAG\\
& UUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAG\\
& GAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUG\\
& GAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCU\\
& UUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUG\\
& GCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCU\\
& UUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUU\\
& GCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAG\\
& UUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAG\\
& GAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUG\\
& GAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCU\\
& UUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUG\\
& GCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCU\\
& UUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUU\\
& GCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAG\\
& UUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAG\\
& GAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUG\\
& GAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCU\\
& UUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUG\\
& GCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCU\\
& UUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUU\\
& GCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAG\\
& UUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAG\\
& GAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUG\\
& GAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCU\\
& UUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUG\\
& GCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCU\\
& UUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUU\\
& GCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAG\\
& UUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAG\\
& GAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUG\\
& GAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCU\\
& UUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUG\\
& GCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCU\\
& UUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUU\\
& GCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAG\\
& UUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAG\\
& GAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUG\\
& \qquad GAGGAGUUUGCUUUGGCUUUGGAGGAGUUUGCUUUGGCUUUGGAGGAG
\end{aligned} \tag{37}
$$

The first four sequences are merely hypothetical, and intentionally small/short, with $\Theta 1_{DNA}$, $\Theta 3_{DNA}$ and $\Theta 4_{DNA}$ being DNA-sequences, while $\Theta 2_{RNA}$ is just some RNA-sequence. $\Theta_{insulin}$ is a DNA-sequence we have already encountered in **Section 2** (see **Equation 6**), while the most verbose sequence we shall encountered in this work is named Θ_{rbcl} and is a real-life, biologically relevant RNA-sequence from a well-known plant protein: **RbcL**[10], the large subunit of **RuBisCO** — the enzyme responsible for carbon fixation in photosynthesis. It is one of the most abundant and essential proteins in plants[14].

6.2 A First Analysis

Before we dive into the matter of solving the analytic problems under the four scenarios introduced in **Section 6.1** above, we shall want to begin with a background analysis that shall give us some high-level insights about any of the sequences in our problem set. In particular, we are going to start by analyzing each of those sequences, so as to know the following basic facts:

1. What is the sequence's symbol set: $\psi(\Theta) =?$
2. Which of the three nucleic acid symbol sets does the sequence belong to — ψ_{DNA}, ψ_{RNA} or ψ_{na}
3. What is the relative frequency of each sequence's symbol-set members? $\forall \rho \in \psi(\Theta), \mu(\rho \in \Theta) =?$
4. How long is the sequence as a flat-structure sequence: $\mu(\Theta) =?$
5. How many codons does the sequence contain? $\Theta \rightarrow \Theta^* : \mathbb{N} \times \psi_{na}^3, \mu(\Theta^*) =?$

For purposes of helping others replicate the analyses and results we shall obtain, we recommend that the following analysis method be used: We can use the TEA[11][28][29] text-processing programming language — by the author, via its `tttt` Linux/Unix package to compute some of the analyses we wish to do via the command-line.

1. **To Process Sequence on Command-Line?** For example, copy-and-paste sequence $\Theta 1$ into a file named `theta1_dna.txt`
2. **To compute the sequence symbol set?** For example, for $\Theta 1$, run the following command against the sequence file to display the unique symbols in the sequence, in their natural order of first-occurrence within the sequence:

```
cat theta1_dna.txt | tttt -c "b:"
TACG
```

[10] Θ_{rbcl} is actually just a sample mRNA Sequence (partial, from *Arabidopsis thaliana* RbcL gene), and is excerpted from a paper that includes the *Arabidopsis thaliana* chloroplast genome, which contains the rbcL gene encoding the large subunit of RuBisCO[27].

[11] Instructions for how to install and use TEA/Transforming Executable Alphabet general-purpose text-processing oriented computer programming language are offered via the project's GitHub: `https://github.com/mcnemesis/cli_tttt`

3. **To compute the modal sequence statistic?** First, note that the **modal sequence statistic** is well introduced, defined and how to use it demonstrated in [6][12]. That said, for example, for Θ1, run the following command against the sequence file to display the unique symbols in the sequence, in their order of most frequent first, and/or their natural order of first-occurrence within the sequence:

```
cat theta1_dna.txt | tttt -c "u!"":"
CGTA
```

Note that the actual command/TEA-code for this is just "u!:", however, because of the way the Linux command-line treats the '!' symbol as a special character — even inside strings, we must cleverly use it in a manner such as shown; we write "u!" ":" to avoid the *"unrecognized history modifier"* error if that code is written directly on the terminal. This shall apply to all the other cases where we must deal with the TEA command qualifier '!"[28] The other workaround is to **first turn off the BASH history expansion modifier**, this, so that we can just write clean TEA code such as "u!:" without errors. So, to disable history expasion for the *active/current session* in your terminal, just run the command:

```
set +H
```

And after that, you can just write clean TEA code for the above task as such:

```
cat theta1_dna.txt | tttt -c u!:
CGTA
```

4. **To compute the distribution of symbols within a sequence?** For example, to compute how many times the symbol 'T' occurs in Θ1, we can merely run the code:

```
cat theta1_dna.txt | tttt -c "d!:T|g:|v:|v!:"
3
```

And for the symbol 'C', we just modify that code to count for 'C' as such:

```
cat theta1_dna.txt | tttt -c "d!:C|g:|v:|v!:"
4
```

But if we wish to do the same for all distinct symbols within the sequence Θ1 — **and especially without having to explicitly list which symbols we are counting within the TEA code**, we could have used "u!:" as in the previous task, since that TEA command counts how many times each unique symbol occurs in a sequence so as to compute the modal sequence, however, currently (with TEA version **1.0.8**[29]), the actual frequencies aren't

[12] See **Definition 1** in [6]

returned with the output of "u!:", and so, we shall want to use a work-around leveraging the above method for computing the frequency of each symbol in the input sequence. We can use the following non-trivial TEA program for that then:

```
cat theta1_dna.txt | tttt -c
 ↪ "v:vSEQ|v:vANA:{}|u!:|v:vMS|l:lMS|y:vMS|d!:^.|v:vSY|y:vMS」
 ↪ |d:^.|v:vMS|y:vSEQ|d*!:vSY|g:|v:|v!:|v:vSYN|g*:{}:vSY:vSY」
 ↪ N|x*:vANA|v:vANA|y:vMS|f:^$:lFIN|j:lMS|l:lFIN|y:vANA"
C4G3T3A2
```

The essential result from running that program against Θ1 is the string **C4G3T3A2**, which tells us that 'C' occurs 4 times, 'G' 3 times, ... , and then 'A' only 2 times.

5. **To compute the cardinality of a sequence?** For example, for Θ2, run the following command against the sequence file to display the total number of all symbols within the sequence:

```
cat theta2_rna.txt | tttt -c "v:|v!:"
18
```

To confirm if TEA is telling us the correct thing, we can also count the characters in that sequence (assuming we pasted just the sequence symbols into the file and nothing else, and no delimiters between them), using the standard Linux/Unix command "wc" as such:

```
cat theta2_rna.txt | wc -c
18
```

Such is the power of the sequence analysis we can do with the TEA language, without any sophisticated tools. Thus, shall we analyze all the other sequences in our problem set, and to simplify things, we shall merely tabulate the analysis results for all six sequences as such:

na-Seq	Unspecific Sequence Symbol Set: $\psi(\Theta)$	Closest Parent Symbol Set	Modal Sequence Statistic, $\overset{>}{\Theta}$ and Symbol Distribution: $\mathbb{u}(\rho \in \Theta)$	$\mathbb{u}(\Theta)$	Codon Count: $\mathbb{u}(\Theta^*) \approx \frac{\mathbb{u}(\Theta)}{3}$
$\Theta 1_{DNA}$	$\langle T, A, C, G \rangle$	ψ_{DNA}	C G T A 4 3 3 2	12	4
$\Theta 2_{RNA}$	$\langle A, U, G, C, T \rangle$	ψ_{na}	G C A U T 6 4 4 3 1	18	6
$\Theta 3_{DNA}$	$\langle A, T, G, C \rangle$	ψ_{DNA}	G A T C 6 5 4 1	16	$5.3\overline{3}$
$\Theta 4_{DNA}$	$\langle T, A, C, G \rangle$	ψ_{DNA}	C G T A 4 3 3 2	12	4
$\Theta_{insulin}$	$\langle A, T, G, C \rangle$	ψ_{DNA}	G C T A 108 105 60 56	329	$109.6\overline{6}$
Θ_{rbcl}	$\langle A, U, G, C \rangle$	ψ_{RNA}	U G A C 1355 1207 305 301	3168	1056

Table 2: A First Analysis of the 5 na-Sequences from **Section 6.1**

6.3 SCENARIO A

`Given some two particular instances of na-Sequences,` $\Theta 1_n$ `and` $\Theta 2_n$`, of the same length,` n`, and with similar symbol sets (`$\psi_{na}(\Theta 1_n) = \psi_{na}(\Theta 2_n)$`), how to quantify how similar or dissimilar they are?`

So, in this scenario, we are basically given some two particular na-Sequences, of the same length, and which have similar symbol sets, and are tasked with how to quantify their similarity or dissimilarity.

Looking at our example sequences above in **Table 2**, only two sequences satisfy the necessary conditions — $\Theta 1$ and $\Theta 4$, which both have length **12** and which have similar symbol sets under base-na[13] — $\psi_{na}(\Theta 1) = \psi_{na}(\Theta 4) = \langle A, C, G, T \rangle$, but also their unspecific sequence symbol sets are already similar.

Actually, this case is very telling concerning what might actually happen with na-Sequences — essentially, we note that, despite the two sequences **actually being different** — for example, $\Theta 1$ ends with codon `TTC`, while $\Theta 4$ ends with `CAT`, and yet, looking at their breakdown analysis via **Table 2**, we see that **they almost seem to be the same!** They have the same sequence symbol set, the same parent symbol set, the same modal sequence and symbol distribution, the same sequence length and thus same number of codons! So, how exactly can we quantifiably distinguish between such sequences in real life?

[13] Formally defined in **Definition 2**

At this juncture, it definitely might make sense to recall what the purpose of the **Anagram Distance Measure**[23] — it essentially is a measure that allows us to tell if any two sequences that might contain the same exact distinct symbols, but possibly in different proportions and/or order, are different or not. So, since we can't use the usually sufficient **modal sequence statistic**[6] to distinguish between these two actually different sequences, let us attempt to compute their anagram distance instead[14].

[14]It shall be **very important** to bring to mind here, the fact that, the cases where comparing sequences or datasets using the ADM might not seem obvious, but as we already saw in a detailed attempt to distinguish between actually different sequences using many various traditional statistical measures (of variation moreover) that all failed to show a difference between some two test sequences — see **Table 1** in [6], one shouldn't take ADM any less serious even for non-numerical data such as analysis nucleic acid sequences!

NOTE:
A Quick Recap **Concerning Interpreting ADM,** $\tilde{A}(\Theta, \Theta^*)$

The Anagram Distance Measure (ADM), associated testing method, background and justifications for the new statistic were first laid out in the seminal paper[23] introducing that measure and its theory. That work was later advanced in the trasformatics paper[6], and essentially, we can note that:

- Given any two sequences, Θ and Θ^*, that ideally should be of the same length, and with the same sequence symbol set, then we compute the anagram distance between them via the formula:

$$\tilde{A}(\Theta \to \Theta^*) = \frac{1}{\text{ɯ}(\Theta)} \times \sum_{i=1}^{\text{ɯ}(\Theta)} |I(\omega_i, \Theta^*) - i| \tag{38}$$

- And that we can interpret the results — which shall always be a positive real number ($\mathbb{R}^+$) as such:

$$\tilde{A}(\Theta \to \Theta^*) = \begin{cases} 0, & \Theta = \Theta^*, \text{no differences} \\ < 1, & \Theta \approx \Theta^*, \text{different in at minimum two positions} \\ = 1, & \Theta \neq \Theta^*, \text{all members shifted by exactly 1 position} \\ > 1, & \Theta \ll \Theta^*, \text{potential indication there is chaos.} \end{cases} \tag{39}$$

For a quick primer, on how this works out, assume we have $\Theta_1 = \langle a, b, c \rangle$, $\Theta_2 = \langle a, c, b \rangle$, $\Theta_3 = \langle b, c, a \rangle$, $\Theta_4 = \langle c, a, b \rangle$, $\Theta_5 = \langle b, a, c \rangle$. Then we can see that:

1. $\tilde{A}(\Theta_1, \Theta_2) = \frac{1}{3}(|1-1| + |2-3| + |3-2|) = \frac{2}{3} < 1$
2. $\tilde{A}(\Theta_1, \Theta_3) = \frac{1}{3}(|1-3| + |2-1| + |3-2|) = \frac{4}{3} > 1$
3. $\tilde{A}(\Theta_1, \Theta_4) = \frac{1}{3}(|1-2| + |2-3| + |3-1|) = \frac{4}{3} > 1$
4. $\tilde{A}(\Theta_1, \Theta_1) = \frac{1}{3}(|1-1| + |2-2| + |3-3|) = \frac{0}{3} = 0$
5. $\tilde{A}(\Theta_1, \Theta_5) = \frac{1}{3}(|1-2| + |2-1| + |3-3|) = \frac{2}{3} < 1$

However, to see the case when $\tilde{A} = 1$, we can use the sequences: $\Omega_1 = \langle a, b, c, d, e, f \rangle$ and $\Omega_2 = \langle b, a, d, c, f, e \rangle$. So, that we have:
$\tilde{A}(\Omega_1, \Omega_2) = \frac{1}{6}(|1-2| + |2-1| + |3-4| + |4-3| + |5-6| + |6-5|) = \frac{6}{6} = 1$
Concerning this last case, it might be worth noting that the only way to have that result for a sequence, is to have each element swapped with its immediate neighbor, without wrapping-around such as we saw happen in Θ_1 Vs Θ_3 for example. Also, the only way this could happen, is if the sequence's cardinality is even.

So, if we return to our na-Sequence analysis, we see that for our tricky case comparing Θ1 Vs Θ4, that if we compute their anagram distance, then we have the results:

$\Theta 4$	T	A	C	T	C	C	G	G	G	C	A	T
$\omega_i \in \Theta 1$	T	A	C	G	G	G	C	A	T	T	C	C
i	1	2	3	4	5	6	7	8	9	10	11	12
$I(\omega_i, \Theta 4)$	1	2	3	7	8	9	5	11	4	12	6	10
$\lvert I(\omega_i, \Theta 4) - i \rvert$	0	0	0	3	3	3	2	3	5	2	5	2

Table 3: A Tabular Analysis of Θ1 Vs Θ4 so as to compute their Anagram Distance

It shall be worth noting concerning how we derive the values of the updated position of ω_i in the compared/resultant sequence, $\Theta 4$, via $I(\omega_i, \Theta 4)$, that we are actually careful to follow the definition of that function (see **Note** on **Page 5** in [6]), with the useful detail that since both sequences contain repeated/duplicated symbols despite their similar length, that we assign to some symbol, such as the first 'T' in $\Theta 1$, the index of the first 'T' in $\Theta 4$, and second 'T' in $\Theta 1$, the index of the second 'T' in $\Theta 4$ — no skipping ahead, no juggling them up[15].

And thus, we can see readily that:

$$\tilde{A}(\Theta 1 \to \Theta 4) = \frac{1}{12}(0+0+0+3+3+3+2+3+5+2+5+2) = \frac{28}{12} = 2.3\bar{3} > 1 > 0 \tag{40}$$

And so, despite the measures and first analysis in **Table 2** having shown that these two nucleic acid sequences didn't have any significant differences, and yet, using ADM as in **Table 3** and **Equation 40**, we find that they are appreciably significantly different! Their ADM being 2.3, it tells us the second sequence potentially has elements in the first sequence shifted to new locations in the sequence, potentially by more than just 1 position. Thus, despite all their other similarities, they do have some differences that we can measure and pin to some telling numbers.

6.4 SCENARIO B

> Given some two particular instances of na-Sequences, $\Theta 1_n$ and $\Theta 2_k$, of different lengths, n and k, but with similar symbol sets ($\psi_{na}(\Theta 1_n) = \psi_{na}(\Theta 2_k)$), how to quantify how similar or dissimilar they are?

So, given the conditions of this scenario and our sample sequences as summarized in **Table 2**, we shall pick sequences $\Theta 1$ and $\Theta 3$, which both have the same sequence symbol set under base-na, and which have cardinalities 12 and 16 respectively.

So, good enough, we have already worked through some of the essential details of the analysis method in **Section 6.3**. However, for this scenario, we can sum up how to conduct our comparative analysis thus:

1. First, conduct tabular analysis so as to see how the two sequences contrast against each other by the dimensions:

   (a) Their unspecific[16] sequence symbol sets.

[15] Thus, we can generally say for any two sequences, Θ and Θ^* under ADM analysis, that for computing the $I(\omega_i, \Theta^*)$ for terms from the source, Θ, even where ω_i occurs multiple times in either sequence, even if at different positions, that we assign the first instance of ω_i, the value of $I(\omega_i, \Theta^*)$ equal to the position of the first instance of ω_i in Θ^*, the second instance of ω_i, the position of the second instance of ω_i in Θ^*, etc. So as to properly/correctly compute the ADM, but also for any other cases in using the **position-index function** in say logic or formulations as we do in many scenarios concerning Transformatics.

[16] The **Unspecific Symbol Set** concept is first introduced in **Definition 4** of the o-SSI paper[18]

(b) Their closest parent/containing/superset symbol set.

(c) Their modal sequence statistic.

(d) Their **sequence characteristic** — more about this soon.

(e) Their cardinality.

We have already seen how to do all the above analysis steps and why, in the First Analysis described in **Section 6.2**.

2. So, if none of the analyses in the first step help show the **quantifiable similarities** or **quantifiable differences** between the two sequences, then proceed to using the more subtle Anagram Distance Measure on them. How to do this we have already covered well in **Scenario A**.
3. In case the ADM also can't find any differences between the two sequences, then most likely, their only differences are **cosmetic** — such as simply naming the same sequence differently, and thus the two sequences under analysis can be considered to be **quantifiably similar**.

So, first, note that, from **Table 2**, that even though the two sequences we are looking at, $\Theta 1$ and $\Theta 3$ have the same **specific sequence symbol set**[17] (under base-na or DNA): $\psi_{na}(\Theta 1) = \psi_{na}(\Theta 3) = \langle A, C, G, T\rangle$, and yet, their **unspecific sequence symbol sets** are different!

So, their first difference is in their unspecific sequence symbol sets: $\psi(\Theta 1) = \langle T, A, C, G\rangle$ and yet $\psi(\Theta 3) = \langle A, T, G, C\rangle$. So, at minimum, we know that they have quantifiable difference in the ordering of their similar symbols within either sequence just by this analysis — especially because these symbol sets respect order of first occurrence within the sequence they summarize.

If we must continue to still analyze the two sequences, we can further note that they have the same closest[18] parent symbol set, ψ_{DNA}. And so, that's a similarity.

And then we come to the matter of their modal sequence statistics. For this case, we see that we have the values:

- $\overset{>}{\Theta}1 = \langle C, G, T, A\rangle$
- $\overset{>}{\Theta}3 = \langle G, A, T, C\rangle$

So, this alone can tell us that the two sequences differ quantifiably in terms of either their relative distribution of members/symbols, or that they differ in the order of first occurrence of their symbols at worst.

Also, note that, the modal sequence statistic (MSS), is definitely computed readily by considering the relative frequency of terms within the sequence under

[17] The concept of the **specific symbol set** is first introduced in **Definition 3** of [18]

[18] We say "closest parent symbol set" because, for several scenarios in analyzing sequences, one can find cases where two sequences have symbols that belong to more than one symbol set where that other symbol set is larger than the sequence symbol set. For example, the case of having to decide if a sequence such as "101" belongs to base 2, base 3, base 10 or even base-36! Or if we have the sequence "AUG" that might span the RNA-symbol set, but also ψ_{na}, and talking of which, we must note that in principle at least, all na-Sequences could also be classified as sequences of terms from the Latin-Alphabet — the symbol set ψ_{az}!

analysis, and so, when we look at the MSS column in **Table 2**, we see that below each of the terms for the MSS, we also display the associated symbol frequencies. This is important as we are to see hereafter...

Talking of which, the next analysis we could conduct, but which is closely related to the previous one, is the **sequence characteristic**. Concerning this, note that, unlike **Scenario A** where the two sequences we analyzed had the same exact MSS, and yet, assuming $\Theta 1$ and $\Theta 4$ had some difference in the frequency of their terms despite having the same MSS, the MSS-analysis alone wouldn't have identified that. And so, before we proceed, let us also introduce/define the sequence characteristic measure.

Definition 7 (The **Sequence Characteristic**, $\hbar(\overset{>}{\Theta})$)**.** *If a sequence Θ has the modal sequence statistic $\overset{>}{\Theta} = \langle \prod_{\omega_i \in \psi(\Theta)} \omega_i, \rangle$, so that we can express it as a string concatenation of its distinct symbols in their relative order of highest frequency and first occurrence such as $\omega_1\omega_2\omega_3 \dots \omega_k$ for some k the size of $\overset{>}{\Theta}$, then, if for each ω_i we also know its corresponding frequency in Θ, such as f_i for ω_i, then writing the frequency next to the symbol such as $\omega_i f_i$ for all terms in $\overset{>}{\Theta}$ produces a string that contains both the information about the unique modal sequence of Θ, but also the information about how frequent each symbol occurred. We shall call such an expression the **Sequence Characteristic**, and shall denote it as $\hbar(\overset{>}{\Theta})$, so that we can then write for Θ, its corresponding sequence characteristic as:*

$$\hbar(\overset{>}{\Theta}) = \prod_{\omega_i \in \psi(\Theta)} \omega_i \cdot f_i \tag{41}$$

Closely related, and since we have already seen such a computation and its application in **Step#4** of our **First Analysis** using the TEA programming language, we might as well formally define a generic machine that can compute $\hbar(\overset{>}{\Theta})$ for any sequence.

Transformer 5 (The **Sequence Characteristic Generator**, gSC)**.**

$$\Theta \xrightarrow{O_{gSC}(\Theta)} \Theta^*; \quad \Theta^* = \hbar(\overset{>}{\Theta}) = \prod_{\omega_i \in \psi(\Theta)} \omega_i \cdot f_i = \prod_{\rho_i \in \overset{>}{\Theta}} \rho_i \cdot \mu(\rho_i \in \Theta)$$

So, we note that, like in the example we saw in generating $\hbar(\overset{>}{\Theta}1)$ for $\Theta 1$ in our **First Analysis** — which returned the characteristic string as **C4G3T3A2**, we can then see that, by using the information in **Table 2**, that for the two sequences we are comparing, we have their sequence characteristics as such:

- $\hbar(\overset{>}{\Theta}1) =$ **C4G3T3A2**

- $\hbar(\overset{>}{\Theta}3) =$ **G6A5T4C1**

And thus, since those two measures aren't the same either, again, we have yet another evidence to conclude the two sequences are quantifiably different.

It should be worth noting, that in contrast to the previous scenario, the sequence characteristic[19] for $\Theta 1$ and $\Theta 4$ are equivalent (as expected?).

6.5 SCENARIO C

> Given some two particular instances of na-Sequences, $\Theta 1_n$ and $\Theta 2_k$, of different lengths, n and k and with different symbol sets ($\psi_{na}(\Theta 1_n) \neq \psi_{na}(\Theta 2_k)$), how to quantify how similar or dissimilar they are?

Without repeating ourselves nor wasting time, note that from the cases we have in **Table 2**, such might be the case with sequences such as $\Theta 1$ and $\Theta_r bcl$ for example. And, from the analysis procedure we have already encountered in **Scenario B**, we can definitely use any of several available ways to quantify their similarities or differences. However, most useful perhaps, might be to just look at their sequence characteristics:

- $\hbar(\overset{>}{\Theta}1) =$ **C4G3T3A2**
- $\hbar(\overset{>}{\Theta}_{rbcl}) =$ **U1355G1207A305C301**

Which, automatically disqualifies any allegations that the two sequences are similar, and which, further tells us they differ in not only their symbol sets, but also in their symbol distribution.

Also, and important to note, we see that, by having a sequence's **characteristic**[20] computed, we can not only tell at a glance which symbols occur most frequent in a sequence, which symbols only occur once but in different order, but also, be able to readily compute the actual cardinality of the summarized sequence — so, with $\hbar(\overset{>}{\Theta}_{rbcl}) =$ **U1355G1207A305C301**, we can compute the length of the **RbcL** by just summing up the f_i terms in $\hbar(\overset{>}{\Theta}_{rbcl})$: 1355 + 1207 + 305 + 301 = **3168**. Thus, the **sequence characteristic** is such a terrific measure when comparing especially large or arbitrarily sized sequences. Good enough, we have seen that there is a simple/short TEA program to automatically compute and display that measure for any input sequence, so, interested researchers and students can just adapt/build on that in the future.

[19] The **Sequence Characteristic** becomes yet another important contribution to the study and analysis of sequences that can serve special purposes than just the related Modal Sequence Statistic, and which, though derived from it, expresses different sequence summarizing information that is important in scenarios such as genetic sequence analysis as we have just seen. Definitely, this measure can, as with other measures we have developed in transformatics, be applied in any mathematical or scientific field and not just statistics or genetics such as in this case.

[20] Yes, sometimes we might just talk of a **characteristic** when we mean a "sequence characteristic".

6.6 SCENARIO D

> How, when given a particular na-Sequence, Q_k, and two collections/samples of other sequences $(\Theta 1, \Theta 2, \Theta 3, ..., \Theta m)$ and $(\Omega 1, \Omega 2, \Omega 3, ..., \Omega n)$ of potentially dissimilar lengths, and that belong to labeled or classified populations --- POPA and POPB, to determine which of the two populations the query sequence, Q_k best-belongs to (so we classify it under one of the labels), as well as determine which of the sample member sequences from either collections it is closest to?

So, unlike the other scenarios we dealt with, seems like the biggest concern here is to compare a sequence not against just one other sequence, but a collection of them — essentially, it's like having to quantify the distance between a particular sequence and some collection of other sequences. So how might we go about solving this using the tools already at our disposal?

NOTE:
Some Useful Ideas Concerning Classification Problems

Though we don't intend to borrow many or any ideas from what others would do or how existing [external and/or independent] methods might provide a solution to our classification problem, we shall call-out one useful general concept that is well placed in the Oxford Dictionary of Computing in relation to this kind of problem:

> **Decision Surface**: A (hyper) surface in a multidimensional state space that partitions the space into different regions. Data lying on one side of a decision surface are defined as belonging to a different class from those lying on the other. Decision surfaces may be created or modified as a result of a learning process and they are frequently used in machine learning, pattern recognition, and classification systems.
>
> — **Oxford Dictionary of Computing**[?]

First, let us deal with the matter of associating our query sequence, Q_k, one of several sequence sets or collections.

6.6.1 Measuring Proximity to a Set of Sequences, Ω^n

So, if we have a set of n different sequences of arbitrary composition and lengths, $\Omega^n = \langle \Omega 1, \Omega 2, \Omega 3, ..., \Omega n \rangle$, then, given some query sequence such as Q_k, and the consequences of **Theorem 2** in [6], we can safely classify the query sequence as **belonging to** or **being appreciably close** to the given population, if we find that computing the anagram distance between the query and the population via their representative modal sequence statistics results in an ADM value of 0, or if, where there are two or more populations to compare against, that we select that population whose ADM against the query sequence's MSS is appreciably close enough to 0.

With that useful theory then, and given we already know how to compute an MSS, $\overset{>}{Q}_k$, for any particular sequence[21] Q_k, we instead better start by solving the

[21] In this work, check **First Analysis**, but also **Section 4.1** and especially **Definition 1** in [6] concerning how to compute the modal sequence statistic.

matter of how to compute an MSS for a set of sequences — a **representative population modal sequence**.

So, assuming we have $Q_k = \langle a, d, c \rangle$, $\Omega 1 = \langle a, b, c, d, e, f \rangle$ and $\Omega 2 = \langle a, b, a, d, c \rangle$, and so that $\Omega^n = \langle \Omega 1, \Omega 2 \rangle$, we wish to compute $\overset{>}{\Omega}^n$, a population modal sequence statistic representing or summarizing the membership and frequency distribution across the two sequences. So, should we go about [correctly] computing $\overset{>}{\Omega}^n$?

STRATEGY 1: Since we have multiple sequences, and that they even have some uncommon terms and dissimilar lengths, a meaningful strategy might be to not attempt to compute the population MSS directly from the individual sequences, and instead use their sequence characteristics. That is, we would compute $\hbar(\overset{>}{\Omega}1)$, $\hbar(\overset{>}{\Omega}2)$, ... etc. and then using these, generate a combined **population characteristic**, $\hbar(\overset{>}{\Omega}^n)$. The essential definition shall help make things clearer:

Definition 8 (The **Population Characteristic**)**.** *Given a collection, Θ^n, of n sequences of arbitrary length and composition. The representative **population characteristic**, that summarizes all the sequences within that population, denoted $\hbar(\overset{>}{\Theta}^n)$, is derived from the sequence characteristics of the contained sequences, $\hbar(\overset{>}{\Theta}_1)$, $\hbar(\overset{>}{\Theta}_2)$, ... , $\hbar(\overset{>}{\Theta}_n)$ as such:*

$$\hbar(\overset{>}{\Theta}^n) = \prod_{\forall \omega_i \in \psi(\Theta^n)} \omega_i \cdot \mathit{u}(\omega_i \in \Theta^n) \tag{42}$$

Where $\forall \Theta_j \in \Theta^n$

$$\mathit{u}(\omega_i \in \Theta^n) = \sum_{\forall \hbar(\overset{>}{\Theta}_j)} f_{\omega_i} \tag{43}$$

And $\forall i, j \in \mathbb{N} : i < j \implies f_i \geq f_j$ for the frequency terms in $\hbar(\overset{>}{\Theta}^n)$.

Thus having computed the representative population characteristic for our sample/collection/set of sequences as $\hbar(\overset{>}{\Omega}^n)$, we then proceed to extract or reduce it to just the **representative population modal sequence statistic** (RPMSS)[22], $\overset{>}{\Omega}^n$.

And with the RPMSS, $\overset{>}{\Omega}^n$, and having computed the MSS of the query sequence, $\overset{>}{Q}_k$, we can then proceed to compute the proximity between the query

[22] Any careful analyst shall realize that computing or deriving the RPMSS as in the given process, or as per **Definition 42**, shall make the most sense, because, any other way would either ignore the important per-sequence symbol distribution information, or might just not be efficient or robust/trustworthy enough, plus, where the frequencies don't matter — such as when each symbol only appears once across all sequences, then working from the per-sequence MSS would help generate a RPMSS that is close-enough to the relative ordering of terms in each member sequence.

and the population via a measure such as the ADM: $\tilde{A}(\overset{>}{Q}_k, \overset{>}{\Omega}^n)$, and if there were more that one population to compare the query against, use the computed ADM values as a **decision surface** to help objectively decide which of the analyzed populations the query most likely belongs to.

Concerning this method too, it is important to note that given there might be cases in which the query sequence and the population contain several uncommon symbols — i.e. $\psi(Q_k) \setminus \psi(\Omega^n) \neq \emptyset$ or that $\psi(Q_k) \cap \psi(\Omega^n) = \emptyset$, we might need to adjust the modal sequences for both the query and population so that we can correctly compute their associated ADM. Thus, we might proceed by eliminating from both $\overset{>}{Q}_k$ and $\overset{>}{\Omega}^n$, those uncommon terms, so that the ADM we use as our decision surface is computed thus: $\tilde{A}(\approx \overset{>}{Q}_k \to \approx \overset{>}{\Omega}^n)$.

EXAMPLE:
How to Compute $\tilde{A}(\approx \overset{>}{Q}_k \to \approx \overset{>}{\Omega}^n)$

Assume we use the given sequences: $Q_k = \langle a, d, c \rangle$, $\Omega 1 = \langle a, b, c, d, e, f \rangle$ and $\Omega 2 = \langle a, b, a, d, c \rangle$, and so that $\Omega^n = \langle \Omega 1, \Omega 2 \rangle$, then:

- $\overset{>}{Q}_k = \langle a, d, c \rangle$
- $\hbar(\overset{>}{\Omega 1}) = a1b1c1d1e1f1$
- $\hbar(\overset{>}{\Omega 2}) = a2b1d1c1$
- $\hbar(\overset{>}{\Omega}^n) = a3b2c2d2e1f1 \quad \Longrightarrow \quad \overset{>}{\Omega}^n = abcdef$

However, given $\overset{>}{\Omega}^n \setminus \overset{>}{Q}_k \neq \emptyset$ and/or $\overset{>}{\Omega}^n \cap \overset{>}{Q}_k = \{a, d, c\}$, then we must adjust as such:

- $\approx \overset{>}{\Omega}^n = acd$

And so that we can then compute $\tilde{A}(\overset{>}{Q}_k \to \approx \overset{>}{\Omega}^n) = \frac{1}{3}(|1-1| + |2-3| + |3-2|) = \frac{2}{3} < 1$.

If there is no other population to compare the query against, we can safely conclude the query is appreciably close enough to the population, otherwise we also compute the ADM between the query and the other populations, and then pick the one with the smallest ADM.

STRATEGY 2: The process outlined in the first strategy to solving the first problem in **Scenario D** might work well and be convincing enough for most practical and theoretical purposes, however, it is not the only plausible route to a good solution. Especially because we are mostly concerned with how to obtain the RPMSS for a collection of potentially widely dissimilar sequences — a very possible case if we are for example dealing with realistic collections of DNA sequence readings/scans of say a person's genome, the genome of some newly identified species with scanty details, or even the case of attempting to obtain an RPMSS for an entire set of individual organisms — e.g human members of a clan or particular family tree, etc. So, even though we might not delve into

it here, there is a proposal to compute $\overset{>}{\Omega}^n$ from n sequences via computing the population's **genome sequence**.

This process would proceed somewhat like this:

- Using the provided collection of sequences Ω^n, and the process outlined in **Section 7.3**, especially via processing such a collection via the **Genome Sequencer** machine/transformer, we shall obtain a representative and summary, well-aligned and potentially complete [na-]sequence Ω_{gs} that represents the entire collection/population provided.
- By the **Identity Genome Sequence Law**, we trust that such a sequence shall be unique for any two distinct collections or populations.
- And thus, having obtained Ω_{gs}, we then compute the RPMSS not from the individual sequences in a population, but instead from their representative genome sequence. That is to say:

  Transformation 15. $\Omega_{gs} \rightarrow \overset{>}{\Omega}_{gs} \approx \overset{>}{\Omega}^n$

  And then we can use the obtained $\overset{>}{\Omega}^n$ and the query's $\overset{>}{Q}_k$ to determine how close they are via an anagram distance measure as we have already seen in earlier examples and analysis.

6.6.2 Measuring Proximity between a Q_k and Members of a Sequence Population, Ω^n

For answering the final aspect of **Scenario D**, we merely can proceed via the following Algorithm:

Algorithm 1 (The **Closest Sequence Algorithm**). *1. Compute the MSS for Q_k — i.e $\overset{>}{Q}_k$.*

*2. Since we are looking for the **sequence closest** to Q_k, then given we know $\overset{>}{Q}_k$, we also know $\psi(Q_k)$.*

*3. Initialize an empty collection Closest Sequence Tuples, **CST** := [].*

*4. Initialize **ADM_CST** := 0.*

*5. Initialize an empty collection Aproximately Closest Sequence Tuples, **ACST** := [].*

*6. Initialize **ADM_ACST** := 0.*

7. For each sequence, Ω_i in Ω^n:

(a) Compute $\psi(\Omega_i)$

*(b) Initialize **ADM_i** := 0.*

(c) If $\psi(\Omega_i) \setminus \psi(Q_k) = \emptyset$:

*i. Compute $\tilde{A}(\psi(\Omega_i) \rightarrow \psi(Q_k))$ and store that in **ADM_i***

ii. Iff $\boldsymbol{ADM\_i} > \boldsymbol{ADM\_CST}$*, add tuple* $[i, ADM_i]$ *to* ***CST****.*

(d) Else/Otherwise:

i. Adjust $\psi(\Omega_i)$ *and* $\psi(Q_k)$ *to eliminate uncommon terms*

ii. Compute $\tilde{A}(\approx \psi(\Omega_i) \rightarrow \approx \psi(Q_k))$ *and store that in* $\boldsymbol{ADM\_i}$

iii. Iff $\boldsymbol{ADM\_i} > \boldsymbol{ADM\_ACST}$*, add tuple* $[i, ADM_i]$ *to* ***ACST****.*

8. If ***CST*** *is not empty:*

(a) Sort ***CST*** *in ascending order of the second term in each contained tuple:* $[i, ADM_i]$ *so that the tuple with the lowest ADM is the first in* ***CST****.*

(b) From Ω^n*, return sequence with index in that topmost tuple as the solution.*

9. Otherwise process ***ACST****:*

(a) Sort ***ACST*** *in ascending order of the second term in each contained tuple:* $[i, ADM_i]$ *so that the tuple with the lowest ADM is the first in* ***ACST****.*

(b) From Ω^n*, return sequence with index in that topmost tuple as the solution.*

□

7 The Mathematics of Genome Sequencing: Sequence Alignment, The Modal Sequence, The Sequencing Machine and the Absolute Identifier Genome Sequence Law

Given what we know of genome sequencing theory and some problems we identified — such as an overall lack of proper mathematical formalism around the issue of how sequencing — or rather, computation of the resultant, consensus and most representative genome sequence (essentially, an identifying **na-Sequence**) of a living organism or particular aspects of it — proteins, mutations, behavioral traits, anomalies or bases of particular or general diseases or incapacities, etc. But even with the matter of mathematically expressing anomalous entities such as viruses and such, wasn't that compelling. And thus this section.

Seeing as the matter of determining the actual na-Sequence that correctly identifies a person or thing might be hinging on the case of attempting to reconstruct a book by mathematically piecing together disparate pieces of it — some in alignment, some not, some supersets of the others, some prefixes or suffixes of others, etc.[23] Also, though we might not immediately surface that philosophy here, the difficulties and problem thus raised, might likewise apply to the matter of trying to computationally "sequence" all of reality or even peculiar small aspects of it — sub-atomic particles that make up living things on the miniature scale, and then planets, stars and galaxies on the grand scale. Is it possible? Does it make

[23]This analogy shouldn't come as a surprise, when compared to how we found it best to describe DNA using a figurative metaphor (see **Introduction**): **Section 1**), but also based on how some authorities seem to be teaching this subject to their students[30].

sense? Is it necessary or even useful? These, and related problems are what we shall attempt to answer in the rest of this section.

7.1 PROBLEM G1: Computing MCS: Maximum Common Subsequence: $\Theta_1 \lozenge \Theta_2$

Problem 1 (*Computing MCS: **Maximum Common Subsequence**).* $: \Theta_1 \lozenge \Theta_2 /$ *Given sequences* $\Theta_1 = \langle k, l, m, o, p, x, y, z \rangle$ *and* $\Theta_2 = \langle b, c, a, y, z, k, l, m, o, p \rangle$*, find the longest common subsequence they share — also to be denoted* $\Theta_1 \lozenge \Theta_2$*, their **M**aximum **C**ommon **S**ubsequence.*

Solution 1. *We shall specify a transformer, tMCS(*Θ_1, Θ_2*) that produces the required solution from the input sequences as such:*

Transformer 6 (The **tMCS** Transformer)**.** $\langle \Theta_1, \Theta_2 \rangle \xrightarrow{O_{tMCS}(\Theta_1, \Theta_2)} \Theta_1 \lozenge \Theta_2;$
$0 \leq \mathscr{u}(\Theta_1 \lozenge \Theta_2) \leq Max(\mathscr{u}(\Theta_1), \mathscr{u}(\Theta_2))$

□

Solution 1 is easier said than done. However, we shall attempt to formally and rigorously express it by formalizing what exactly the $tMCS(\Theta_1, \Theta_2)$ transformer does. Basically, we are to compute the MCS via **Algorithm 2**.

Algorithm 2 (The **tMCS** Algorithm)**.** *1. Initialize the global maximum common sequence,* O_g^m*, to the empty set.*

2. *Given **a collection of** n **sequences**,* $\Theta^n = \{\Theta_1, \Theta_2, ...\}$*.*
3. *Start by computing the cardinality of each sequence,* $\Theta_i \in \Theta^n$*, and note the **largest sequence cardinality**,* $Max(\mathscr{u}(\Theta_i) \forall \Theta_i \in \Theta^n) = Max(\mathscr{u}(\Theta_i))$*, for the longest sequence,* Θ_m *such that* $Max(\mathscr{u}(\Theta_i)) = \mathscr{u}(\Theta_m)$*.*
4. *Compute the cardinality of a sequence that can exactly contain the given sequences each concatenated to the other, but with the longest sequence duplicated. Essentially, compute:*

$$C_R = \mathscr{u}(\Theta_m) + \sum_{\forall \Theta_i \in \Theta^n} \mathscr{u}(\Theta_i) \tag{44}$$

5. *Sort the collection of sequences,* Θ^n*, in ascending order of the sequence cardinality. Let the sorted sequence collection be denoted* $\hat{\Theta}^n$
6. *Start with the first two sequences from* $\hat{\Theta}^n$*, such as* Θ_1 *and* Θ_2*, and place them within two adjacent arrays* $A_1^{C_R}$ *and* $A_2^{C_R}$ *of equal length* C_R *— for simplicity, we shall just denote them as* A_1 *and* A_2*, in such a way that* Θ_1 *occupies in its containing array* A_1*, positions from 1 to* $\mathscr{u}(\Theta_1)$*, while* Θ_2 *occupies positions in the range:* $[(1 + \sum_{\Theta: \mathscr{u}(\Theta) > \mathscr{u}(\Theta_2)} \mathscr{u}(\Theta)) = I(\Theta_2[1] \in A_2)$[24]*,* $I(\Theta_2[1] \in A_2) +$

[24] We write $\Theta_2[1]$ to mean the first element in Θ_2 — i.e. position 1 has value 1, and the nth position has value n, not usual $n-1$

$\text{ш}(\Theta_2) - 1]$, *which also means,* $\forall a_i \in \Theta_j$ *such that* $I(\Theta_j, \hat{\Theta}_n) < I(\Theta_2, \hat{\Theta}_n)$, *it implies* $j < I(\Theta_2[1] \in A_2)$.
And Equivalently, the range $[\text{ш}(\Theta_1) + 1, \text{ш}(\Theta_1) + 1\text{ш}(\Theta_2) - 1] = [\text{ш}(\Theta_1) + 1, \text{ш}(\Theta_1) + \text{ш}(\Theta_2)]$.
Visually, we can express this as:

A_1	$a_{1,1}$	$a_{1,2}$	$\cdots$	$a_{1,\text{ш}(\Theta_1)}$						
A_2					$a_{2,1}$	$a_{2,2}$	$\cdots$	$a_{2,\text{ш}(\Theta_1)}$		

7. *Update the two sequence alignment arrays as such:*
   (a) *Keep the first array,* A_1, *unchanged (since it contains the longest sequence,* $\Theta_1 \in \hat{\Theta}_n$*).*
   (b) *Set* $j = z$, *where* z *is the* ***shift step size*** *— simplest scenario,* $z = 1$.
   (c) *Shift each element in the second sequence by* z*-steps to an earlier position in array* A_2, *so that the two sequences* Θ_1 *and* Θ_2 *have an overlap across the alignment arrays/matrix in positions spanning range:* $[\text{ш}(\Theta_1) + 1 - j, \text{ш}(\Theta_1)]$, *and so that* Θ_2 *shall be occupying the updated range:* $[\text{ш}(\Theta_1) + 1 - j, \text{ш}(\Theta_1) + \text{ш}(\Theta_2) - j]$
   (d) *Scan the overlapping sections in both sequences, in positions* $[\text{ш}(\Theta_1) + 1 - j, \text{ш}(\Theta_1)]$, *and by comparing each item in that range — i.e.* $\Theta_1[i]$ *Vs* $\Theta_2[i]$, *determine the longest common subsequence, or rather the* ***MCS****, thus:*

   Assume the overlap is of length m *such that the overlap can be visually depicted as such:*

$\approx \Theta_1$	$a_{1,k}$	$a_{1,k+1}$	$\cdots$	$a_{1,k+m}$
$\approx \Theta_2$	$a_{2,l}$	$a_{2,l+1}$	$\cdots$	$a_{2,l+m}$
$(\Theta_1[i] == \Theta_2[i])?$	*1*	*0*	$\cdots$	*1*
$\approx (\Theta_1[i] == \Theta_2[i]) \approx O^m$	$f(a_{1,k}, a_{2,l})$	*0*	$\cdots$	$f(a_{1,k+m}, a_{2,l+m})$

   for some $k \in [1, \text{ш}(\Theta_1)]$ *and* $l \in [1, \text{ш}(\Theta_2)]$.
   Create another array of maximum length m, O^m, *which contains at most* m *elements generated thus:*
   - $O^m[i]$ *contains 0 if the corresponding two elements in* Θ_1 *and* Θ_2 *didn't match. Otherwise it contains the matching element value from the first sequence — or rather,* $f(a_{1,k}, a_{2,l}) = a_{1,k}$ *iff* $a_{1,k} == a_{2,l}$.

- *truncate O^m to its longest non-zero/non-empty subsequence, set that as O^m.*
- *Iff $\mathcal{u}(O^m) > \mathcal{u}(O_g^m)$, then **Update** or replace O_g^m with O^m $\implies$ $O_g^m := O^m$*

(e) Update $j \implies j := j + z$

(f) Iff $j = \mathcal{u}(\Theta_2)$, then return O_g^m

*(g) Otherwise Iterate from **Step#c***

7.2 PROBLEM G2: Computing Overlap Resultant Sequences: $\Theta_1 * \Theta_2$

Problem 2 (*Computing ORS: **Overlap Resultant Sequence***)**.** *: $\Theta_1 * \Theta_2$] Given two overlapping/well-aligned sequences Θ_1 and Θ_2 of the same length but potentially different terms at any position in the range $[1, \mathcal{u}(\Theta_1)]$, how to compute a resultant sequence $\Theta_1 * \Theta_2$ that is most meaningful for DNA sequences?*

Solution 2. *Assuming $\Theta_1 = \langle \prod_{i=1}^{n} a_i, \rangle$ and $\Theta_2 = \langle \prod_{i=1}^{n} b_i, \rangle$, construct Θ^* via the following tORS transformer:*

Transformer 7 (The **tMCS** Transformer)**.**

$$\langle\langle \Theta_1 \langle \prod_{i=1}^{n} a_i, \rangle, \Theta_2 \langle \prod_{i=1}^{n} b_i, \rangle\rangle \xrightarrow{O_{tORS}(\Theta_1,\Theta_2)} \Theta_1 * \Theta_2;$$

$$\mathcal{u}(\Theta_1 * \Theta_2) = \mathcal{u}(\Theta_1) = \mathcal{u}(\Theta_2) = n$$

$$\wedge \quad \Theta^* = \langle \prod_{i=1}^{n} f(a_i, b_i), \rangle$$

*for $f(x, y)$ **a resolver function** defined as such:*

$$f(x,y) = \begin{cases} x, & x == y, \text{terms are the same} \\ x, & x = \neg(y), \text{one term is the complement of the other} \\ y & \text{otherwise.} \end{cases}$$

□

7.3 PROBLEM G3: Computing the Genome Sequence (GS): $(\Theta_1 \lozenge \Theta_2)^*$

Problem 3 (*Computing GS: **Genome Sequence***)**.** *: $(\Theta_1 \lozenge \Theta_2)^*$] Given any two sequences, Θ_1 and Θ_2, compute their **resultant genome sequence**, denoted $(\Theta_1 \lozenge \Theta_2)^*$, that contains at core, their longest common subsequence (the MCS), $O^m = (\Theta_1 \lozenge \Theta_2)$, potentially padded either side by the specially computed prefix and suffix subsequences from either input sequence as such:*

$$(\Theta_1 \lozenge \Theta_2)^* \quad = \quad \Theta_1^a \cdot (\Theta_1^b * \Theta_2^b) \cdot (\Theta_1 \lozenge \Theta_2) \cdot (\Theta_1^c * \Theta_2^c) \cdot \Theta_2^a \tag{45}$$

And where the special terms are derived and named as such:

- Θ_1^a *is the non-overlapping section of* Θ_1 *before the* ***COS****.*
- $(\Theta_1^b * \Theta_2^b)$ *is the consensus sequence from* Θ_1 *and* Θ_2 *such that its elements are the result of applying the* ***resolver function*** *to compute* $f(\Theta_1[i], \Theta_2[i])$ *as specified in* ***Transformer 7****. It occurs* ***before the MCS****.*
- $(\Theta_1 \Diamond \Theta_2)$ *is the* ***maximum common subsequence****, MCS,* O_g^m*, of* Θ_1 *and* Θ_2*. It is the heart of the computed genome sequence.*
- $(\Theta_1^c * \Theta_2^c)$ *is the consensus sequence from* Θ_1 *and* Θ_2 ***after the MCS****.*
- Θ_2^a *is the non-overlapping section of* Θ_2 *after the* ***COS****.*

COS*,* $\overline{\Theta_1\Theta_2}$*, which is the* ***Consensus Overlapping Sequence****, padded on either by the non-overlapping sections of either* Θ_1 *or* Θ_2*, has the following properties:*

- $\overline{\Theta_1\Theta_2} = (\Theta_1^b * \Theta_2^b) \cdot (\Theta_1 \Diamond \Theta_2) \cdot (\Theta_1^c * \Theta_2^c)$
- $\Theta_1 \approx \Theta_1^a \cdot \overline{\Theta_1\Theta_2}$
- $\Theta_2 \approx \overline{\Theta_1\Theta_2} \cdot \Theta_2^a$

So that we could rewrite ***Equation 45*** *as such:*

$$(\Theta_1 \Diamond \Theta_2)^* \quad = \quad \Theta_1^a \cdot \overline{\Theta_1\Theta_2} \cdot \Theta_2^a \tag{46}$$

Finally, we know that the cardinality of the genome sequence, $\mathbb{u}((\Theta_1 \Diamond \Theta_2)^*)$*, is bounded thus:*

$$\mathbb{u}(\Theta_m) \leq \mathbb{u}((\Theta_1 \Diamond \Theta_2)^*) \leq \mathbb{u}(\Theta_m) + \sum_{\forall \Theta_i \in \Theta^n} \mathbb{u}(\Theta_i) \tag{47}$$

As per ***Equation 44*** *and the sensibilities of GTNC[16].*

7.4 The Genome Sequencer

Definition 9 (A **Genome Sequencer**). *A solution to **Problem 3**, is a special sequence processor, $\tilde{T}(\Theta^n)$, that can compute and return a resultant sequence, R_{Θ^n}, also known as the **genome sequence**, defined as*[a]

$$R_{\Theta^n} = (\Theta_m \Diamond \Theta^n \setminus \Theta_m)^* \tag{48}$$

Such that Θ_m is the longest sequence in Θ^n — a collection of n sequences, and that $\tilde{T}(\Theta^n)$ produces/generates/computes R_{Θ^n} by iteratively computing and updating the genome sequence in n-1 iterations as such:

Transformation 16. $R^0_{\Theta^n} \xrightarrow{O_{tGS}(\{R^0_{\Theta^n}\} \cup \Theta^n, 0)} R^1_{\Theta^n} \xrightarrow{O_{tGS}(\{R^1_{\Theta^n}\} \cup \Theta^{n-1}, 1)} R^2_{\Theta^n} \xrightarrow{O_{tGS}(\{R^2_{\Theta^n}\} \cup \Theta^{n-2}, 2)} \dots \xrightarrow{O_{tGS}(\{R^j_{\Theta^n}\} \cup \Theta^{n-j}, j)} R^{j+1}_{\Theta^n} \dots \xrightarrow{O_{tGS}(\{R^{n-2}_{\Theta^n}\} \cup \Theta^1, n-2)} R^{n-1}_{\Theta^n}$

*To produce the final genome sequence, $R^{n-1}_{\Theta^n}$ via the **tGS Transformer** defined as such:*

Transformer 8 (The **tGS Transformer**, $\tilde{T}(\Theta^n)$). $R^j_{\Theta^n} \xrightarrow{O_{tGS}(\{R^j_{\Theta^n}\} \cup \Theta^{n-j}, j)} R^{j+1}_{\Theta^n}$; $R^0_{\Theta^n} = \emptyset$
and $R^{j+1}_{\Theta^n}$ is the resultant genome sequence after processing Θ^n with the first j sequences removed from it, and with the previous genome sequence, $\{R^j_{\Theta^n}\}$ appended to it, so as to produce that next genome sequence $R^{j+1}_{\Theta^n}$.

[a] Note that, even though in our formalism here we express the **relative complement set** of $\{\Theta_m\}$ in Θ^n as $\Theta^n \setminus \Theta_m$ — especially to keep the notation simple, we actually mean $\Theta^n \setminus \{\Theta_m\}$, which means, the first set with the elements in the second set omitted.

That transformer, $\tilde{T}(\Theta^n)$, is a genome sequencer, and is essentially a transformer that reduces a matrix of na-Sequences to a resultant sequence, R_{Θ^n}, that is the resultant genome sequence of a specie, population, or any natural entity represented by or expressed by the [potentially incomplete or approximate] genome sequences in the collection Θ^n.

7.5 The Identity Genome Sequence (IGS) Law

Law 3 (The **Identity Genome Sequence Law**). *The Identity Genome Sequence, $R_{\Theta^n}(\Omega) : \mathbb{N} \times \psi_{DNA}$, derived from some collection of sequences, $\Theta^n \langle \prod^n \theta_i \rangle$ such that $\theta_i \subset \Omega \vee \theta_i \approx \Omega$ are approximations of Ω, the longest known genome sequence of the entity, whose exact genome sequence would be Ω, and which can uniquely identify it. For sufficiently large n, and where the sample sequences θ_i were read or generated correctly by scanning or sequencing the entity, then*

$$R_{\Theta^n}(\Omega) \approx \Omega_m \tag{49}$$

Where Ω_m is the known best approximation of the genome sequence that correctly identifies any instance of a member of the kind Ω.

NOTE:
On **Consequences of IGS**

One potentially significant consequence of **Law 3** is that every distinct living thing — unique or distinct animal, plant, eukaryote or prokaryote, has a distinct identifying genome sequence, Ω, that we can discover and/or approximate via significantly many readings of particular or representative [sub-]sequences of the DNA of the observable expressions of Ω.
However, given that in reality it might not be exactly possible to exhaustively compute the correct value of R_{Θ^n}, nor tell exactly when that approximation approaches Ω exactly, it makes one wonder whether we can ever accurately know what it is that exactly underlies the expression of our reality — living things or not[a].

[a]Of course, there is the queer possibility that a kind of genome sequencing might be applicable to also inanimate/non-living things! But that's a philosophical discussion for another day.

8 The Modal Sequence Statistic as a Generic Genetic Code — A Case of Genome Expression in Bio-Automata

```
Genome Expression: The collective expression profile of all genes
within a genome, including coding and non-coding regions, across
different conditions, cell types, or developmental stages. It
is a systems-level view --- measuring how the entire genome behaves
dynamically. This includes: mRNA expression of all genes, non-coding
RNA activity, epigenetic regulation, chromatin accessibility and
transcriptional networks. In essence, gene expression is local,
while genome expression is global.
```

— **Microsoft Copilot**[14]

Away from using transformatics to make sense of the differences between organisms based on their genetic code sequences, an even more exciting application of the theory and practical methods of transformatics would be in conceptualizing, analyzing and explaining the important biological, physical, chemical, informational and mathematical concept of **genome expression**. We shall use a thought experiment and hypothetical cases leveraging exemplary genetic code modeled using special sequences that are treated as modal sequences at minimum.

First, we shall define a basic cipher that can encode the basic digits (of base-10), as some unique, but non-digit forms (essentially, with non-digit *glyphs*). This, so when we speak of a living organism — for example a pine-apple or a feline, we don't expect that in nature, or rather, that by physically dissecting the organism, that one shall find trapped or stored inside it, the literal genetic code symbols (such as those in ψ_{na}), but rather, that they shall find natural literal expressions or instances of genetic code material basis — such as the chemical base molecules

that define nucleobases, or nucleotides, and also complex molecular structures such as transcribed amino-acids post-RNA translation of DNA expressions.

So, in our hypothetical genome expression model, we are going to expect elements from our basic hypothetical DNA symbol set:

$$\psi_{\Omega} = \langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \rangle \equiv \psi_{10} \tag{50}$$

Which, especially for curiosity's sake, but also, for creative and expressive reasons, is meant to allow us to explore a peculiar genetics model based on not just some four base symbols as is the case in ψ_{DNA}, but which allows us to look at any base-10 number expression as a potential genetic code sequence with **each digit expressing a distinct hypothetical nucleic acid**. For example, we might describe some fictitious organism — for simplicity's sake, perhaps just a virus that we shall name **the Euler Virus** or just "veuler", as having its entire genome sequence encoded as just:

$$\Omega_{veuler} = 27182818 \tag{51}$$

However, that is the DNA-side of the story. As in real nature, we need a *different* alphabet or symbol set for expressing genetic code in its *intermediate form* — as genome expression typically proceeds via expression of DNA into mRNA first of all, and then finally into functions such as proteins. So, in our case, the intermediate expressions shall be expressed using the **Ozin Cipher**, which we are to describe hereafter.

8.1 The Ozin Genetic Code Cipher

With the background we have obtained thus far, note that the special **transcription level** expression of our genetic code originally expressed via ψ_{10}, shall be transcribed into the intermediate **ozin genetic code** that spans the symbol set ψ_{oz} that is mapped from the genetic code storage expression via a mapping as depicted in **Figure 2**.

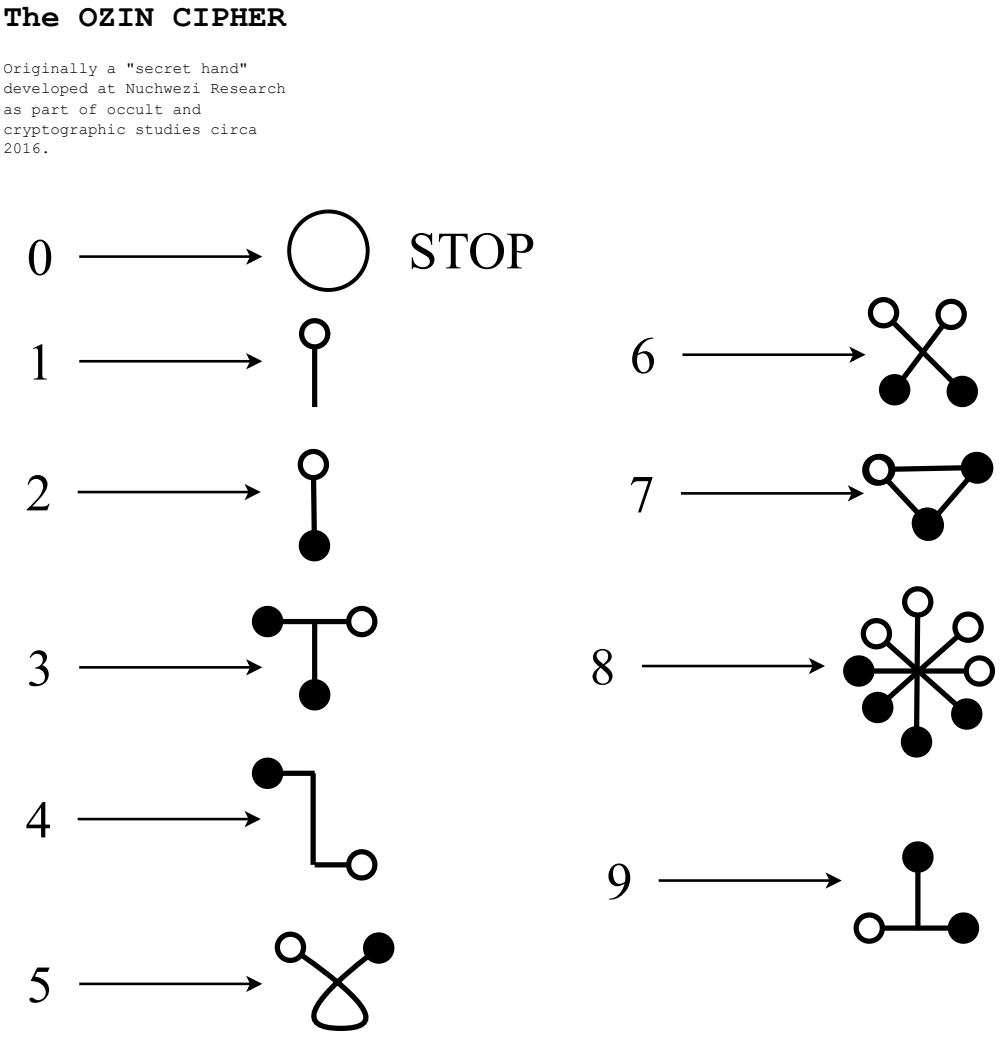


Figure 2: The OZIN Cipher mapped from Decimal Symbols.

For simplicity's sake, we might interpret or transcribe from base-Ω DNA into our base-OZ RNA expression as such:

If for some organism such as that expressed via $\Omega_{veuler} = 27182818$, we wish to encode the equivalent intermediate [physical] expression, then we iterate through each symbol in the source code sequence, and re-write it as the equivalent symbolic structure in OZIN, and this, so as to center the ozin glyphs along a **backbone structure** of just a mere line that starts with a tiny dot and ends with the last ozin structure from the source sequence transcribed. Essentially, for our Euler Virus, it would render as something of the sort shown in **Figure 3**

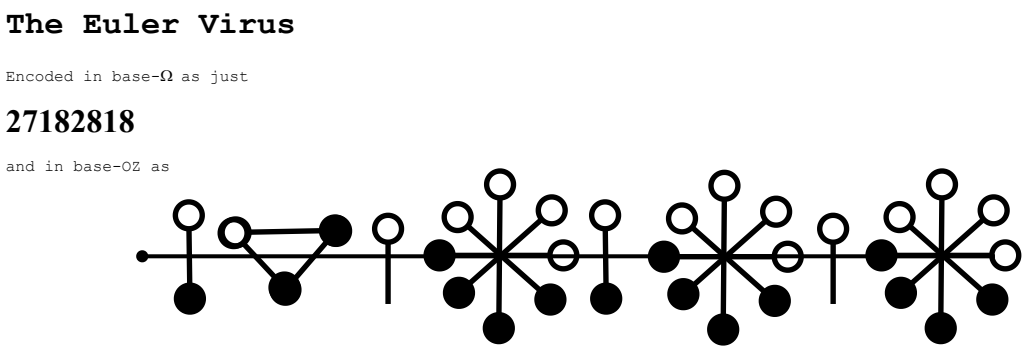


Figure 3: The equivalent OZIN expression of the hypothetical Euler Virus genome sequence.

Another organism, which we might just refer to as the **Hifinelle**, is actually based off of our favorite base-10 o-SSI — the **Hi-Fi o-SSI**[23], and thus, its corresponding genetic code at rest is as:

$$\Omega_{Hifinelle} = 8649137520 \tag{52}$$

And which, after we have it transcribed into intermediate OZIN genetic expression, shall appear as shown in **Figure 4**

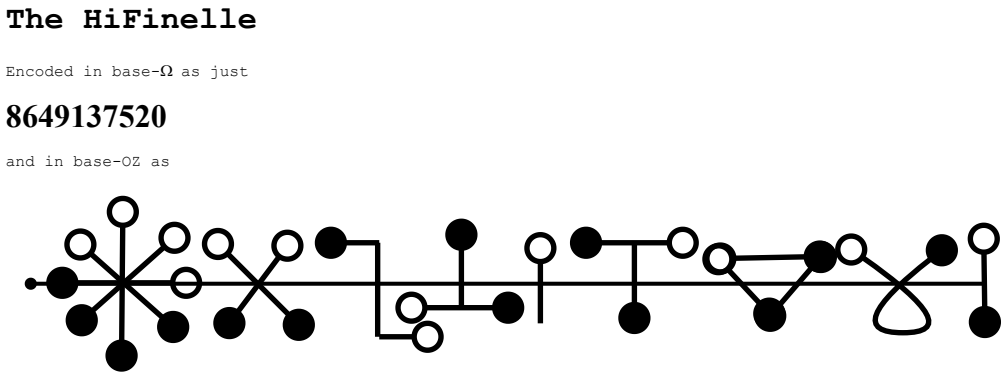


Figure 4: The equivalent OZIN expression of the HiFinelle genome sequence.

Of course, in our simplistic genome expression system, the occurrence of that last "0" symbol in the HiFinelle genome sequence tells us to STOP or just exclude that symbol with a GAP in case any other symbols follow thereafter. It is our STOP-codon equivalent.

Another genome sequence,

$$\Omega_3 = 0123026 \tag{53}$$

Might help illustrate that point — see **Figure ??**

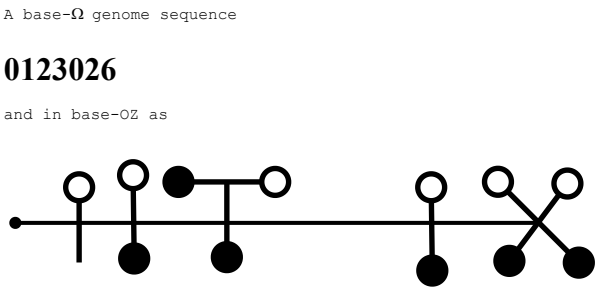


Figure 5: The equivalent OZIN expression of the Ω_3 genome sequence.

9 Gene Expression in Living Organisms Leveraging Genetic Code (DNA $\rightarrow mRNA \rightarrow Protein \rightarrow$ Organism)

`Code: A rule for transforming a message from one symbolic form (the source alphabet) into another (the target alphabet), usually without loss of information. The process of transformation is called encoding and its converse is called decoding.`

— **The Oxford Companion to the Mind**[2]

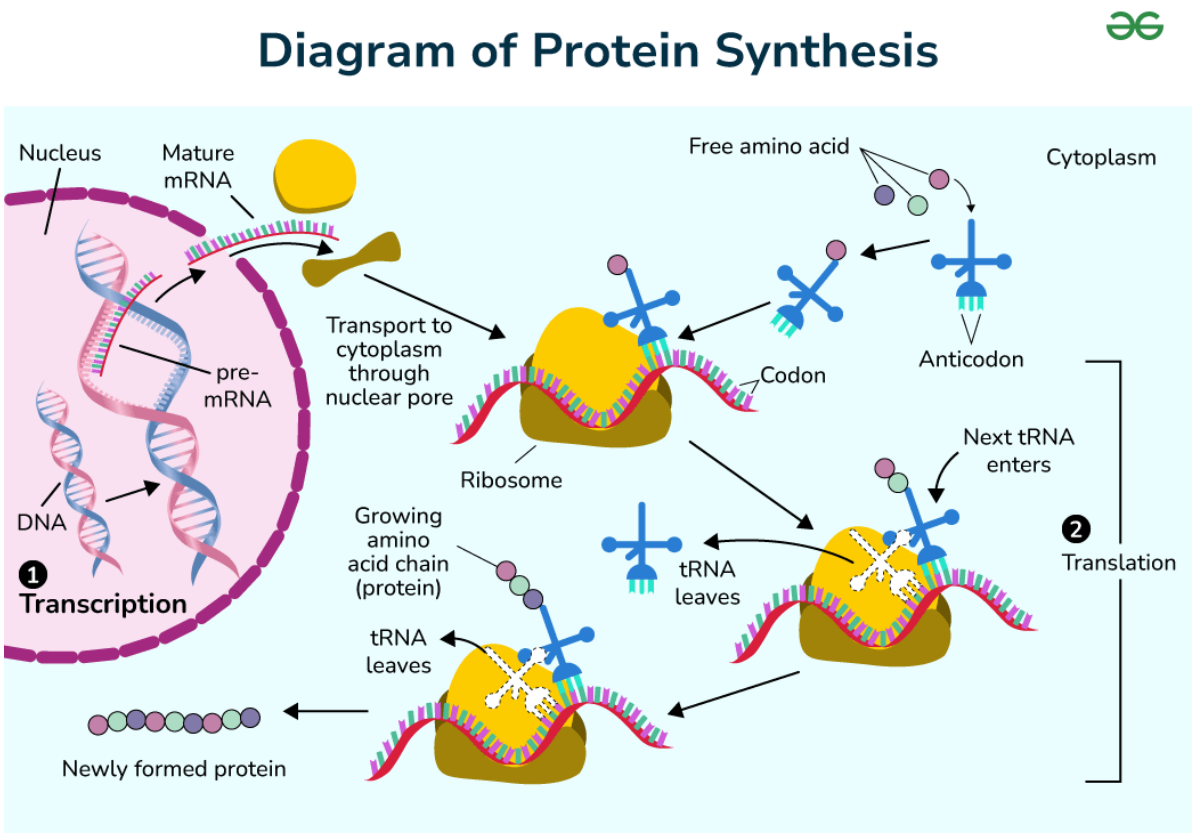


Figure 6: An illustration of the gene translation process in a cell via protein-factories known as ribosomes[31]

In the previous section we have dealt with genome expression, and have especially dealt with hypothetical systems and used the conceptual modal sequence concept as the basis for exploring expression and/or manifestation of an entire organism from just its basic genetic code. In this section though, we shall focus on actual expression in real and natural biological systems, and shall focus more on expression at a molecular level — essentially, at the level of protein manifestation via **gene expression**.

It shall be interesting to note that for genetic code in natural automata (nature-like bio-automata[25] and generally living things), the most important reason the genetic coding language exists, is so that the body/host-organism system can produce required materials as and when they are needed or demanded for. This for example means producing new or extra body tissue in a still growing organism or in one with any damaged or missing tissue, and essentially, such productions are about the synthesis of particular molecules in the body's system that are basically proteins. The diagram in **Figure 5** is a basic illustration of the process for living organisms — eukaryotes especially.

[25]My research assistant — thanks **Microsoft Copilot**, did bring it to my attention that the term "bio-automata" is "usually reserved for models or conceptual representations of biological systems — especially those designed to simulate behaviors, growth patterns, or decisions-making processes using predefined rules, like in cellular automata or agent-based modeling." And thus, much as I often find it attractive to use the term — as an umbrella term including actual living organisms which, from the perspective of the computer scientist in me, are still correctly classifiable under the "biological automata" category in my opinion since they actually operate on some infallible inherent natural program in their DNA. But, I shall adhere to the advise of my assistant for now.

For simplicity's sake, we can assume the following summarization of the basic process that fully and correctly breaks down the typical ordeal:

First, we shall assume that given the protein is just a chain of amino-acids, we might as well just think of it as though it were an ordered sequence of some terms, and thus, in keeping with the notation from transformatics, we might just refer to such a protein with our usual typical **resultant sequence** symbol — Θ^*.

And so, given that these proteins are actually nearly direct/1-to-1 mappings from the corresponding DNA sub-sequence code of a finite length, we might then refer to the DNA sequence that encodes the instructions for producing Θ^* with just the basic typical transformatics **source sequence** symbol: Θ — more conveniently, because we wish to also talk of the length of the sequence, we might preferably write the DNA sequence code of length **n** (meaning for example, **it contains exactly n DNA-codons**), as Θ_n. So, for example we might more fully express Θ_n as such:

$$\Theta_n = \langle a_{ij}, \rangle; a_{ij} \in \psi_{DNA^*} \quad \forall j \in [1, n], i \in [1, 64] \quad \wedge \quad n \in \mathbb{N} \tag{54}$$

Equation 53 being just a sometimes preferable way to write the same exact sequence as:

$$\Theta_n = \langle a_1, a_2, a_3, ..., a_i, ..., a_{n-1}, a_n \rangle; \forall i \quad \exists a_i \in \psi_{DNA^*} \quad \wedge \quad n \in \mathbb{N} \tag{55}$$

We have defined the special symbol sets ψ_{DNA^*} in **Section ??** and ψ_{DNA} in **Definition 1**, and as for ψ_{DNA^*}, we know that it essentially is the set of the distinct 64 codons (see **Table 1** in [2]) that *especially* encode amino acids, and which were first introduced in **Section 1**. Another way to expound on this is by saying that:

$$\Theta_n = \{a_i \quad | \quad a_i = \prod_{\rho \in \psi_{DNA}}^{3} \rho \quad \wedge \quad \forall i \in [1, n], n \in \mathbb{N} \quad \wedge \quad \psi_{DNA} = \langle A, C, G, T \rangle\} \tag{56}$$

Thus we might encounter a gene such as Θ_4 composed of exactly 4 codons as shown below:

$$\Theta_4 = \langle \langle A, T, G \rangle, \langle A, A, A \rangle, \langle T, T, A \rangle, \langle T, A, G \rangle \rangle \tag{57}$$

Which, might also equivalently be expressed as a flattened sequence if the fact that the nucleobases it contains are always read in triplets/3-grams/tuples of 3 at a time. So that we can merely write it as:

$$\Theta_{4\times 3} = \langle A, T, G, A, A, A, T, T, A, T, A, G\rangle \tag{58}$$

So we imply that under the flat-structure notation, the sequence has exactly 4×3 elements. By this fact and the observation that the previous nested notation merely helps to group together each codon's members within a meaningful sub-sequence, and that the order is otherwise maintained in the flat-sequence structure, we might then also equivalently express the same actual DNA code sequence as an $n \times 3$ matrix as shown below:

$$\Theta_{4\times 3} = \begin{pmatrix} A & T & G \\ A & A & A \\ T & T & A \\ T & A & G \end{pmatrix} \tag{59}$$

Whether to actually express it as a $3 \times n$ matrix or as $n \times 3$ might be up to the particular taste of the mathematician or scientist, but otherwise, we know that the ordered sequence Θ in any of the forms above is essentially a **genetic program** to guide a DNA code processor such as a ribosome construct a corresponding protein based on the equivalent transcribed mRNA code sequence Θ_4^* written in mRNA code as such:

$$\Theta_4^* = \langle\langle A, U, G\rangle, \langle A, A, A\rangle, \langle U, U, A\rangle, \langle U, A, G\rangle\rangle \tag{60}$$

Which is what we would obtain after a necessary **DNA** $\rightarrow$ **mRNA** transform attainable via a DNA sequence transformer we might define as such:

Transformer 9 (DNA to mRNA Encoder). $\Theta_n \xrightarrow{O_{mRNA-encode}(\cdot)} \Theta_n^*$;
$\forall a_i \in \Theta_n = \langle a_i, \rangle : n, \quad a_i \in \psi_{DNA} \equiv \{A, T, C, G\}$,
and if $\exists a_i \in \Theta_n : a_i = T \implies \exists a_i^* \in \Theta_n^* : a_i^* = U$
Otherwise $a_i = a_i^* \implies \text{ɯ}(a_i = T \in \Theta_n) = \text{ɯ}(U \in \Theta_n^*) \quad \wedge \quad \text{ɯ}(\Theta_n) = \text{ɯ}(\Theta_n^*) = n.$

i	a_i	**Amino Acid (Code-name)**	**Function**
1	ATG	Methionine (Met)	Start Codon: initiates translation
2	AAA	Lysine (Lys)	Basic amino acid
3	TTA	Leucine (Leu)	Non polar amino acid
4	TAG	Stop (Amber)	Terminates translation

Table 4: Amino-Acid Codes and Names in Θ_4, a DNA-encoded gene

Thus, though the gene in its DNA form comprised of the ordered sequence of amino-acid codes named as in **Table 4**, and yet, the resultant sequence after applying **Transformer 9** to Θ_4 would be as explained in **Table 5**.

i	a_i	**Amino Acid (Code-name)**	**Function**
1	AUG	Methionine (Met)	Start Codon: initiates translation
2	AAA	Lysine (Lys)	Basic amino acid
3	UUA	Leucine (Leu)	Non polar amino acid
4	UAG	Stop (Amber)	Terminates translation

Table 5: Amino-Acid Codes and Names in Θ_4^*, a mRNA encoded gene

So, note that the names (and code-names) of the mRNA encoded codons stay the same as those of their corresponding DNA-encoded codons in both tables — this is actually generally/conventionally so. But also, note that the functions of the individual codons in either scenario are likewise expressed the same. So, this is is because, when the genetic code is actually being executed (such as in standard protein-synthesis), the processor (the ribosome) merely operates on the mRNA-encoded gene and not directly on the original DNA-encoded code sequences.

Also, important to note, the processor only produces an amino acid (as part of the protein synthesis program), only after having encountered a "start" instruction, and we know that such instructions are the kind encoded by **start codons**, of which the most universally utilized START-codon is **ATG/AUG**

known as Methionine, but also other rare-scenario[26] START-codons include the mRNA codes GUG and UUG — used as such in prokaryotes, and then AUU and AUA, used as such in humans only.

And then, the processor will stop the protein construction task once it encounters a gene instruction of the "stop" kind. These are encoded using the **stop codons**, and these are strictly any one of: TAA/UAA (Ochre), TAG/UAG (Amber) and TGA/UGA (Opal)[32].

That said, further note that, after processing the gene, and/or after encountering a stop-codon, the ribosome (also understood as the "protein factory") is then triggered to detach (from the "assembly line") and then release/return the final assembled protein thus far. These resultant proteins are basically just a chain of **actual amino acids** generally starting with the Met-amino acid.

And then, further note that, in case any codons were encountered before the AUG (or rather *start-codon*), these shall then be merely be skipped — they aren't processed or won't translate into any product such as the usual case of producing an amino-acid (this, even if they would normally have triggered the production of some amino acid).

9.1 Protein Manufacturing Algorithm

So, overall, we might sum up this critically important protein generation process with a convenient formalism such as with a protein-production algorithm expressed as in the flow-chart depicted in **Figure 6**.

[26] They are used less frequently, mostly in prokaryotes and some organelles. When used as START codons, they still recruit the initiator tRNA and translate as **methionine**, not their usual amino acid[14]

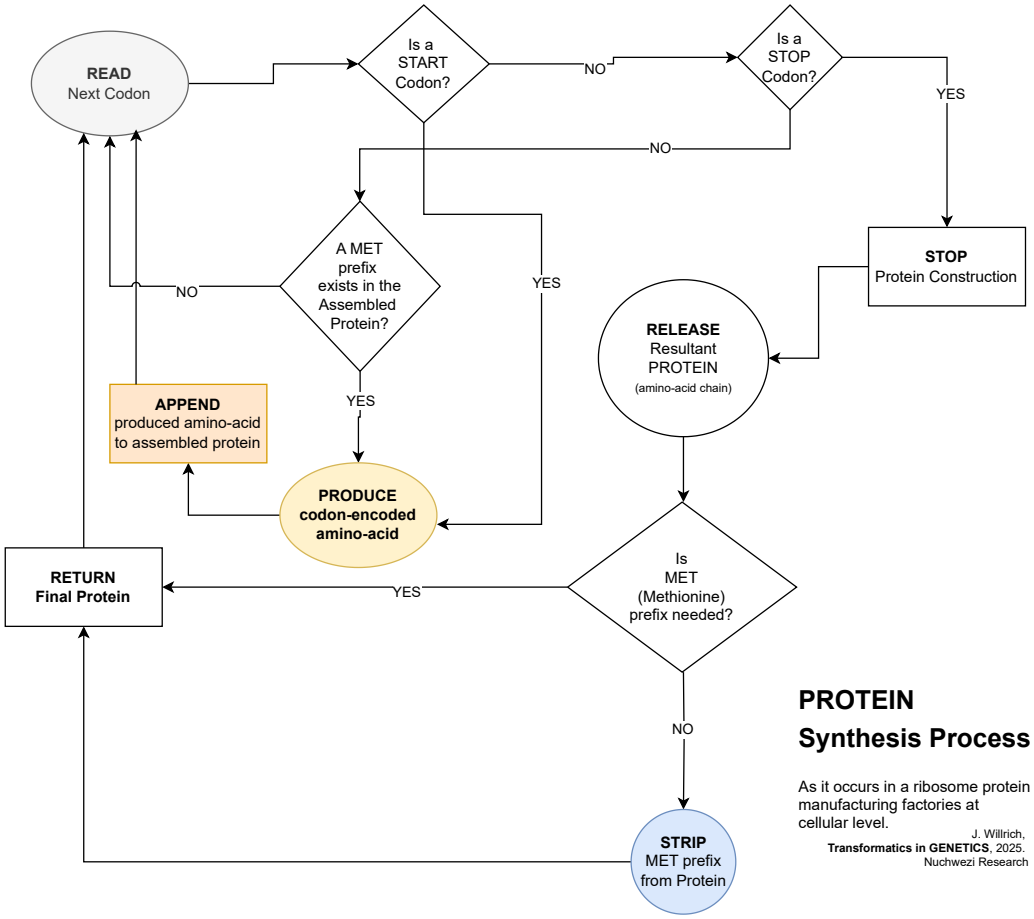


Figure 7: **Flow-Chart summarizing the Ribosome-based Protein-Synthesis Process in a Living Cell**

That process which is depicted in **Figure 6** is how the ribosome protein-manufacturing factory operates at a cellular level as depicted using a **Flow Chart Diagram** — meaning, the states of the operating environment as well as those of the operator (the ribosome) are interlaced with decision-making scenarios so as to bring to mind the logic behind how the process proceeds. However, in a different diagram — the **Ribosome State Machine** as depicted in **Figure 7**, we clearly abstract everything else away and focus on what actually happens from the point-of-view of the gene code sequence processor — the ribosome. In a way, that state machine not only depicts the various states the ribosome shall be in while operating on incoming gene-code sequences (kind of *requests for solutions/solution-instances/proteins* to problems/specifications/genes) and then how it goes about producing the out-going amino-acid sequences (the proteins). It might even start to feel like the ribosome is a kind of 3D-printer, which, when presented with the specifications of a particular 3D-sequence, knows to process it (translate it) and then produce the required/specified object that is in the context

of biological systems we are looking at here, essentially proteins[27].

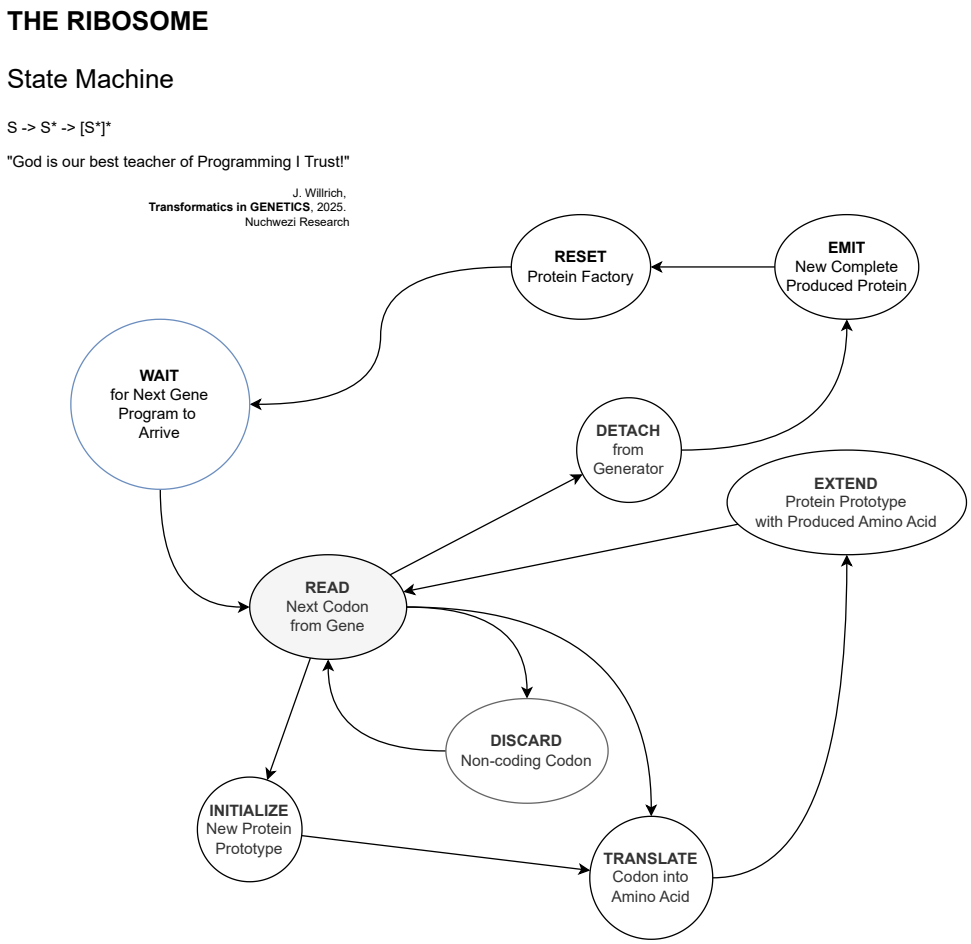


Figure 8: **The RIBOSOME State Machine**

Before concluding this section on biological computing systems of the sequence generator kind[35], it should be worthwhile noting that, as depicted in **Figure 7**, the process that is naturally found in every living thing's tiniest cells (excluding viruses as already saw in **Section 1**, could be, as with any legitimate state machine, be used to implement a proof-of-concept ribosome processor (or a *ribosome computer* — a way to implement bio-automatons that behave like a ribosome or which process code similar to DNA and mRNA, or which generally operate on sequences to produce other (possibly more complex) sequences. Thinking of a robotic ribosome might not be something that is immediately required in contemporary medicine or in-organism computing systems, but might be a concept worth adapting or exploring for the design and implementation of self-contained robotic factories for manufacturing and outputting complex products producible via use of a sequence-processing, assembly-line kind of method just as proteins

[27]Of course, for someone with a background in computer science and who also has interest in designing not just computer programs but also new kinds of computers - abstract machines or not, studying the ribosome from a computing theory and computer architecture perspective — such as so one appreciates what Instruction Set the ribosome employs; how the ribosome compares to say a Von-Neumann architecture machine; is it Turing Complete or not? How might a pure-text processing languages such as TEA[33] implement a ribosome simulation in say a web-browser environment?[34] or that failing, at least allow for the creation of a protein generator or even an entire organism generator as a simulation of how gene code sequences can be translated into sequences of multi-dimension objects?[35] etc.

are produced in a bio-cell by the ribosome. We might for example think of "a sequence from which a finished particular type of car can be manufactured at will" or "a sequence from which certain kinds of sequence-form/sequence-based[28] artificial and/or organic creatures or bots might be systematically produced at will or on-demand". These kinds of printers — which, unlike just printers of things on paper, or 3D-printers that know to only produce plastic variants of models they are fed with, but which can say *print a human*[29], teleport a cow or a banana, etc. might be interesting to explore, as we look into the far-future, where, with humanity's ability to travel and survive in remote and/or unnatural worlds away from their biological home environments (such as Earth's biosphere), might compel them to have to develop new kinds of machines that would not only print out ideas on paper, but also complete food ready-to-eat, medicines to particular kinds of ailments, certain kinds of companion creatures or species, etc. Means to survive on/in alien worlds by leveraging smart, general sequence processors and generators. Talking of which, the next section shall help us start to appreciate this perspective of using the case of genetic code sequences and the ribosome sequence processor into the dimension of both artificial as well as conceptual or hypothetical bio-machines that like the bio-cell can produce complex things via processing of some kinds of code sequences. Bio-automata.

10 Gene Expression in Bio-Automata Leveraging Genetic Modal Sub-Sequences (Numero-Gene Code → Ozin-Gene Code → Platonic-Form Organelles)

One might begin by wondering: **using the concepts from transformatics and pure mathematics, how might we model or express a general and realistic ribosome?** A plausible and meaningful solution to this problem would be to begin by acquiring or developing *a rigorous and correct working definition of what a ribosome is.* The answer would follow directly from that, thus our first definition in this section; a formal definition of a ribosome in any system natural or artificial:

Definition 10 (A **Ribosome**)**.** *Assuming we re-write a sequence of DNA code in terms of the ordered sequence of nucleotides it contains, as in* ***Equation 55*** *re-written as in* ***Equation 60****:*

[28] Vertebrates anyone(?)

[29] Though we might touch on it in a future paper on philosophy — e.g in *Computational Mysticism*[36], it might be interesting to air-out the author's illuminating view that unlike most other bio-mata such as beasts in the wild or fish in the seas, and definitely not as with silicon-based automatons such as GPT-powered modern *disincarnate* artificial entities — nor the likes of the ZHA qAGI[37], that the human in particular, has this peculiar attribute to them that, apart from just their material substratum as any physical robot might possess or need — and which is say the domain of "material producers" such as the ribosome is, and away from their conceptual/software substratum too, they seem to, or perhaps arguably, also possess a preternatural layer of existence that perhaps is or might not have anything to do with their DNA/material-blueprint. A better or more precise classification term for such [*preter-*]intelligent -mata/matter(?) might be the still underground term and concept of a *Psymaton* or **Psymata** — bits of this line of discourse have been already touched on in the Psymaz Interview[38].

$$\Theta_n = \{a_i \quad | \quad a_i = \prod_{\rho \in \psi_{DNA}}^{3} \rho\} : n \tag{61}$$

*Θ_n would then be any sequence of DNA-codons of length n, equivalent to an equivalent flat-structure ordered sequence of nucleotides of length $n \times 3$. We can then produce mRNA-codons from Θ_n as per **Transformer 9**, so that we produce a new mRNA-codon sequence of length n that is generated as such:*

Transformation 17. $\Theta_n \xrightarrow{O_{mRNA-encode}(\cdot)} \Theta_n^*$;

$\Theta_n^* = \{a_i^* \quad | \quad a_i^* = \prod_{\rho \in \psi_{mRNA}}^{3} \rho\} : n$

And with Θ_n^* produced, we can then merely generate the corresponding ordered sequence of amino-acids, denoted as $[\Theta_n^*]^*$, via the following mRNA to amino-acid transformer:

Transformer 10 (mRNA to Amino-Acid Translator). $\Theta_n^* \xrightarrow{O_{mRNA-translate}(\cdot)} [\Theta_n^*]^*$;

1. *$\text{ɯ}([\Theta_n^*]^*) < \text{ɯ}(\Theta_n^*)$ because we only count each codon in the source once, and as per the rules of gene processing/transcription, all non-coding codons (introns) — basically, codons not able to be translated into an amino-acid given the state of the gene processor[30], don't contribute to the generated resultant sequence in terms of sections it contains — with the exception of the special "Met" codon[31] that might or might not be retained in the resultant sequence even though it is automatically included as the first produced amino-acid in any legitimate gene sequence.*
2. *The entire sequence $[\Theta_n^*]^*$ is a kind of 3-Dimension molecule based on the chain of amino-acids it contains, and is technically referred to as a protein.*

□

*A **Ribosome** then, is any combination of transformers that can result in $[\Theta_n^*]^*$ when presented with just Θ_n as per the two intermediate transformer processes **Transformer 9** and **Transformer 10**, and whose overall processing algorithm is as depicted in **Figure 6** and its corresponding state machine as in **Figure 7**.*

*So, overall, a ribosome is any machine that can implement the combined transformer defined as in **Transformer 11**:*

[30] See **Figure 6** and **Figure 7**

[31] A START-codon also counted among genuine "exons" — coding codons

Transformer 11 (The **Protein Generator** (A Ribosome)). $\Theta_n \xrightarrow{O_{mRNA-encode}(\cdot)} \Theta_n^* \xrightarrow{O_{mRNA-translate}(\cdot)} [\Theta_n^*]^*$;
$\mathcal{U}([\Theta_n^*]^*) < \mathcal{U}(\Theta_n^*) = \mathcal{U}(\Theta_n) = n :$
$\psi_\Theta = \psi_{DNA} \quad \wedge \psi_{\Theta_n^*} = \psi_{mRNA} \quad \wedge \psi([\Theta_n^*]^*) = \psi_{amino-acids}$

And thus, we can finally merely call any machine capable of implementing the protein generator in ***Transformer 11*** *as a* ***Ribosome****.*

11 Conclusion

In this paper, we have advanced our knowledge concerning...

Applying TRANSFORMATICS

“A woman is a string that is a replicating enclosure. You invest a substring of your string in her for a while, and then she transforms it into a higher string which she eventually kicks out into the world. The woman is a very special transformer, the man mostly a generator. That’s most of natural biology expressed using TRANSFORMATICS.

Another way to look at it; Tukaruga ha musaana, twaija omunsi, yatuzaara.. kyakweta okutufoora. Baitu emara netubinga kugenda omumwanya kugarukayo... Kikuzooka niho KITARA.

So, it’s perhaps merely humbling to see how a basic mathematics originally meant to explain artificial intelligence and abstract machines, automatons, can likewise explain or express the creative, dynamic science of biological systems as well as the queer idea that life is an investment by the sun into planetary ecosystems, and which can later escape their closures to further express it in distant realms such as in remote galaxies and space-times!”

— a reflection about the new mathematical field of TRANSFORMATICS
as applied to explaining various natural and artificial phenomenon.

Foundation Paper: https://bit.ly/transformatics101

fut. prof. J. Willrich
(currently at Nuchwezi Research)

References

[1] Carl Sagan. *Cosmos*. Book Club Associates, London, UK, 1981. Special edition for **Book Club Associates**.

[2] Richard L. Gregory and Oliver L. Zangwill, editors. *The Oxford Companion to the Mind*. Oxford University Press, Oxford, UK, 1987. Available online at https://archive.org/details/oxfordcompaniont00greg.

[3] BioExplorer.net. Do bacteria have nucleus? https://www.bioexplorer.net/do-bacteria-have-nucleus.html/, 2025.

[4] Seungho Kang, Alexander K Tice, Frederick W Spiegel, Jeffrey D Silberman, Tomáš Pánek, Ivan Čepička, Martin Kostka, Anush Kosakyan, Daniel M C Alcântara, Andrew J Roger, et al. Between a pod and a hard test: The deep evolution of amoebae. *Molecular Biology and Evolution*, 34(9):2258–2270, 2017. https://academic.oup.com/mbe/article/34/9/2258/3827454.

[5] OpenStax Biology. Viruses - biology libretexts. https://bio.libretexts.org/Bookshelves/Introductory_and_General_Biology/General_Biology_1e_(OpenStax)/5:_Biological_Diversity/21:_Viruses, 2025.

[6] Joseph Willrich Lutalo. **The Theory of Sequence Transformers & their Statistics**: The 3 information sequence transformer families (anagrammatizers, protractors, compressors) and 4 new and relevant statistical measures applicable to them: Anagram distance, modal sequence statistic, transformation compression ratio and piecemeal compression ratio. *Academia.edu.*, 2025. https://doi.org/10.6084/m9.figshare.29505824.v3.

[7] Wikipedia contributors. Base pair. https://en.wikipedia.org/wiki/Base_pair, 2025. Accessed August 2, 2025. Includes diagram of DNA double helix showing A–T and C–G base pairing.

[8] Grady Venville and Jenny Donovan. Analogies for life: a subjective view of analogies and metaphors used to teach about genes and dna. *Teaching Science*, 52(1):18–22, 2006. Available at: https://research-repository.uwa.edu.au/en/publications/analogies-for-life-a-subjective-view-of-analogies-and-metaphors-u.

[9] University of Nebraska–Lincoln. Complementary, antiparallel dna strands — dna and chromosome structure. https://passel2.unl.edu/view/lesson/6f214d098527/4, n.d. Accessed July 29, 2025.

[10] Genomics Education Programme. Where does our genome come from? https://www.genomicseducation.hee.nhs.uk/education/core-concepts/where-does-our-genome-come-from/, 2025. Accessed July 29, 2025. Explains how sperm and egg each contribute half the genome, forming a unique combination in the zygote.

[11] OpenStax Biology. Gametogenesis (spermatogenesis and oogenesis). https://bio.libretexts.org/Bookshelves/Introductory_and_General_Biology/General_Biology_1e_(OpenStax)/43:_Animal_Reproduction_and_Development/43.3C:_Gametogenesis_(Spermatogenesis_and_Oogenesis), 2025. Accessed July 29, 2025.

[12] University of Leicester. The cell cycle, mitosis and meiosis for higher education. https://le.ac.uk/vgec/topics/cell-cycle/the-cell-cycle-higher-education, n.d. Accessed July 29, 2025.

[13] Shinichi Mochizuki. The geometry of frobenioids i: The general theory. https://www.kurims.kyoto-u.ac.jp/~motizuki/The-Geometry-of-Frobenioids-I.pdf, 2008. Accessed August 2, 2025. Introduces Frobenioids as symbolic categorical structures encoding arithmetic transformations.

[14] Microsoft Copilot. Clarifying discussions on dna sequence facts and genetics in general during drafting manuscript. AI-generated insights via Copilot discussion, 2025. Personal communication, July 2025.

[15] Wikipedia contributors. Nucleic acid sequence. https://en.wikipedia.org/wiki/Nucleic_acid_sequence, 2025. Accessed July 2025.

[16] Joseph Willrich Lutalo. A general theory of number cardinality. *Academia.edu*, Jan 2024. *Accessible via* https://www.academia.edu/43197243/A_General_Theory_of_Number_Cardinality.

[17] Nature Education. The four bases – atcg. https://www.nature.com/scitable/content/the-four-bases-atcg-6491969/, n.d. Accessed July 2025.

[18] Joseph Willrich Lutalo. Concerning a special summation that preserves the base-10 orthogonal symbol set identity in both addends and the sum. *Academia*, 2025. *Accessible via* https://www.academia.edu/download/122499576/The_Symbol_Set_Identity_paper_Joseph_Willrich_Lutalo_25APR2025.pdf.

[19] Regina Bailey. Genetic code and rna codon table. https://www.thoughtco.com/genetic-code-373449, 2019. Accessed July 2025. Explains RNA nucleotide composition and codon structure.

[20] National Center for Biotechnology Information (NCBI). Refseq: Ins homo sapiens insulin [nm_000207.2]. https://databases.lovd.nl/shared/refseq/INS_NM_000207.2_codingDNA.html, 2020. Accessed July 31, 2025.

[21] J. Craig Venter et al. The sequence of the human genome. *Science*, 291(5507):1304–1351, 2001. Available at: https://www.science.org/doi/pdf/10.1126/science.1058040.

[22] S. Anderson, A. T. Bankier, B. G. Barrell, et al. Sequence and organization of the human mitochondrial genome. *Nature*, 290:457–465, 1981. Available at: https://www.nature.com/articles/290457a0.pdf.

[23] Joseph Willrich Lutalo. Introducing the anagram distance statistic, Ã, a quantifier of lexical proximity for base-10 o-ssi and any arbitrary length ordered sequences, its relevance and proposed applications in computer science, engineering and mathematical statistics. *Academia.edu*, 2025. *Accessible via* https://doi.org/10.6084/m9.figshare.29402363.

[24] Joseph Willrich Lutalo. Concerning debugging in tea and the tea software operating environment. *Academia.edu*, 2025.

[25] Joseph Willrich Lutalo. Philosophical and mathematical foundations of a number generating system: The lu-number system. *Academia.edu*, 2025. *Accessible via* https://doi.org/10.6084/m9.figshare.29262749.

[26] Wikipedia contributors. Dna and rna codon tables. https://en.wikipedia.org/wiki/DNA_and_RNA_codon_tables, 2025. Accessed August 2, 2025. Provides codon-to-amino acid mappings and genetic code/name tables.

[27] Shusei Sato, Yasukazu Nakamura, Takakazu Kaneko, Erika Asamizu, and Satoshi Tabata. Complete structure of the chloroplast genome of arabidopsis thaliana. *DNA Research*, 6(5):283–290, 1999.

[28] Joseph Willrich Lutalo. Tea taz - transforming executable alphabet a: to z: Command space specification. https://doi.org/10.6084/m9.figshare.26661328, 2024.

[29] mcnemesis. cli_tttt: Command line interface for tttt. https://github.com/mcnemesis/cli_tttt/, 2024. Accessed: 4th Aug, 2025.

[30] Teresa Przytycka. Lecture 10: Whole genome sequencing and analysis. https://www.ncbi.xyz/CBBresearch/Przytycka/download/lectures/PCB_Lect10_Whole_Genome.pdf, 2025. Accessed August 2, 2025. Lecture slides from Introduction to Computational Biology.

[31] GeeksforGeeks. Diagram of protein synthesis. https://www.geeksforgeeks.org/biology/protein-synthesis-diagram/, 2024. Accessed July 29, 2025. Shows transcription, translation, and ribosome-mediated protein synthesis.

[32] Susha Cheriyedath. Start and stop codons. https://www.news-medical.net/life-sciences/START-and-STOP-Codons.aspx, 2019. Accessed July 29, 2025. Describes canonical and alternative start codons across organisms.

[33] Joseph Willrich Lutalo. Software language engineering-text processing language design, implementation, evaluation methods. *Preprints*, 2024. *Accessible via* https://www.preprints.org/frontend/manuscript/3903e4cd075074a7005cb705a5ef26c5/download_pub.

[34] Joseph Willrich Lutalo. Tea research: Tea on the web a high-level web software operating environment specification for the tea programming language: Web tea architecture. https://doi.org/10.6084/M9.FIGSHARE.29591687, n.d. figshare.

[35] Joseph Willrich Lutalo. Applying transformatics: Sequence generators. https://doi.org/10.6084/M9.FIGSHARE.29654645, 2025. FigShare.

[36] Joseph Willrich Lutalo. Pragmatic computational mysticism. https://doi.org/10.6084/M9.FIGSHARE.27187071, 2024. figshare.

[37] Joseph Willrich Lutalo. Introducing zha, a real q-agi. *FigShare*, 2025. *Accessible via* https://doi.org/10.6084/M9.FIGSHARE.29049794.

[38] Joseph Willrich Lutalo. Unraveling mysteries of the zha q-agi chatbot: an interview by icc, of fut. prof. jwl and m*a*p ade. psymaz of nuchwezi. *FigShare*, 2025. *Accessible via* https://doi.org/10.6084/M9.FIGSHARE.29064671.

Applying TRANSFORMATICS In GENETICS

(STILL JUST A PAPER)

- ✓ COMPUTING GENETIC PROXIMITY VIA ADM
- ✓ GENETIC ANALYSIS WITH N-GRAM SUBSEQUENCES
- ✓ DNA AS MODAL SEQUENCE STATISTICS
- ✓ GENE EXPRESSION VIA CODE TRANSFOMERS

& MORE!

JOSEPH WILLRICH LUTALO