Brief instructions for using the programs of the package phonetic-languages-simplified-examples-array

Oleksandr Zhabenko

November 8, 2021

Author and software developer: Oleksandr Zhabenko

Introduction

There are different languages. They have different structure and rules. It is possible to create and use (based on one of existing widely used and well-spoken languages, in particular Ukrainian in this work) a 'phonetic' language that is better suited for poetry and music. It is even possible to create different versions of phonetic language. This paper proposes to create several different phonetic languages based on Ukrainian.

Imagine that you can understand the information in the text regardless of the order of the words and preserve only the most necessary grammar (for example, the rule does not separate the preposition and the next word is preserved). Understand just like reading a text (after some learning and training, perhaps), in which only the first and last letters are preserved in words in their positions, and all the others are mutually mixed with each other. So imagine that you can understand (and express your thoughts, feelings, motives, etc.) the message of the text without adherence to strict word order. In this case, you can organize the words (keeping the most necessary grammar to reduce or eliminate possible ambiguity due to grammar, or rather a decrease in its volume), placing them so that they provide a more interesting phonetic sounding. You can try to create poetic (or at least a little more rhythmic and expressive) text or music.

It can also be an inspiring developmental exercise in itself. But how could you quickly find out which combinations are more or less fit? Also, can the complexity of the algorithms be reduced?

These are just some of the interesting questions. This work does not currently provide a complete answer to them, but is experimental one and a research, and any result of it is valuable.

Ukrainian is a language without strict requirements for word order in a sentence (although there are some established preferred options) and has a pleasant sounding. So, it can be a good example and instance. In addition, it is a native language for the author of the programs. Even if you don't want to create and use 'phonetic' languages where phonetics is more important than grammar, then you can assess the phonetic potential of the words used in the text to produce specially sounded texts. It can also be valuable and helpful in writing poetry and possible other related fields [43].

Sound Representations Durations as the Basis of the Approach

There is the fact as the basis of the approach that the language sounds have different durations, which depends on the many factors e. g. mean of the phoneme producing (the different one for every one of them), other factors that can be more or less controlled but usually the full control is not required and is not achieved. This leads to the fact that chaining of the phonemes and phonetic phenomenae sequentially among which there is also their syllables groupping introduces some rhythmic painting (picture, scheme). A human can (that is also trainable and can be developed) recognize the traits of the picture, compare them one with another, come to some phonetic-rhythmic generalizations and conclusions.

The question of determining the duration of speech sounds is not easy, but the exact result as already mentioned is not required. In this implementation of the approach of phonetic languages, certain statistical characteristics of sounds are used, in particular, possible durations are determined. If we compare the method of determining durations, which is proposed and used in the program of the r-glpk-phonetic-languages-ukrainian-durations package, the analogy will be the packaging of bulky objects. For the observer, the packaging will be an imaginary model of the process of obtaining sound durations. The pldUkr program (its generalization pldPL from the phonetic-languages-phonetics-basics package also follows this path, but it does not have normalization, because for different languages there may not be such a phenomenon as palatalization) uses linear programming to find the minimum convex hull (not in a strict mathematical sense), which can 'contain' the sounds of speech. This convex hull has an analogy of packaging, while the sounds of speech have an analogy of objects of variable volume inside the package. The same sound can be used in different situations, in different words with different durations, but the program tries to choose such durations that would 'cover' (similar to the envelope curve 'covers' a family of curves) all these variations for all sounds, with an additional normalization of the duration of the phonetic phenomenon of palatalization (softening) of the consonant, which is least controlled by man, and therefore it is expected that this duration is the most resistant to possible random or systematic fluctuations. For the Ukrainian language the possible duration which does not change strongly sounding is defined experimentally with use of the computer program mm1.

Finally, normalization is not mandatory, it is important that all durations are proportional to each other, i.e. it is not the durations themselves (which are numerically expressed as real positive numbers) that are important, but their mutual ratios (it is allowed to multiply these durations simultaneously by one and the same positive number that does not affect the results of the approach).

Polyrhythm as a Multi-Ordered Sequence Pattern

Let us have some sequence organized in the following way. Let us implement (generally speaking a conditional one) division of the sequence into compact single-connected subgroups with the same number of elements each in the subgroup, which actually means that we split the sequence into a sequence of subsequences with the same number of elements in each. Consider the internal ordering of each subsequence from the perspective of the placement of the values of its elements and repeatability of the some patterns of the placement of the elements. We assume that the elements can be compared in relation of order, that is, they are the elements of the data type that has an implemented instance of the class Ord.

Considering that the elements of the subsequences may be pairwise different (or in some cases equal), we will compare the positions on which the subgroups of elements that have a higher degree of relatedness (”closeness”, ”similarity”) in value and order are located. Denote such subgroups by indices that have in the module code mostly a letter designation.

Then each subsequence will consist of the same number of elements of one nature (in particular, numbers of the type Double), in each subsequence there will be selected several subgroups of ”similar” elements in value (and order, if the subsequences are sorted by the value), each of which will have its own index as a symbol (most often in the code – the characters). Subgroups must have (actually approximately) the same number of elements (in the code it is not strictly used for simplification of the former one, but it is so in the vast majority of cases because of the excessive ”accuracy” of numbers of type Double that are used). Consider the question of positions in the subsequences of the corresponding subgroups in case of they have been belonging to different subsequences.

To assess this, we introduce certain numerical functions that have regular behavior and allow us to determine whether the subsequences actually have elements that belong to the relevant corresponding subgroups in the same places, or on different ones. It can be shown that the situation ”on different ones” corresponds to the presence of several rhythmimc patterns - for each subgroup will be their own, which do not mutually match, at the same time the ideal situation ”completely in the same places” corresponds to the case when these rhythms are consistent with each other, as is the case of coherence in quantum physics, in particular spatial and temporal coherence, which is important in particular for understanding of lasers and masers. Polyrhythms consisting of such rhythms, which cohere with each other, form a more noticeable overall rhythm, as well as the presence of coherence in the radiation leads to a more structured latter one [45].

As an illustration for the ideas of the section the following data.

An example of the rhythmic sequence (almost ideal case).

```
Prelude Rhythmicity.PolyRhythm Numeric> let f x = putStrLn . showFFloat (Just 4) (sin (2*pi*x)) $ ””
   in mapM_ f [0,0.2..4]
```

```
6

0.0000
0.9511
0.5878
-0.5878
-0.9511
-0.0000
0.9511
0.5878
-0.5878
-0.9511
-0.0000
0.9511
0.5878
-0.5878
-0.9511
-0.0000
0.9511
0.5878
-0.5878
-0.9511
-0.0000

Prelude Rhythmicity.PolyRhythm Numeric> getPolyChRhData 'a' 5 (PolyCh [True,True,True,False] 5)
  (PolyRhythm [1,1,1,1,1]) . map (sin . (*pi) . (*2)) $ [0,0.2..4]
[[RP P c,RP P a,RP P b,RP P e,RP P d],[RP P c,RP P a,RP P b,RP P e,RP P d],[RP P c,RP P a,
  RP P b,RP P e,RP P d],[RP P c,RP P a,RP P b,RP P e,RP P d]]
```

Here is the example of the (weak-)non-rhythmic sequence.

```
Prelude Rhythmicity.PolyRhythm Numeric> let f x = putStrLn . showFFloat (Just 4)
  (sin (27182.81828459045*pi*x)) $ "" in mapM_ f [0,0.01..0.24]
```

```
0.0000
-0.5139
-0.8817
-0.9988
-0.8319
-0.4284
0.0969
0.5947
0.9233
0.9894
0.7742
0.3388
-0.1930
-0.6698
-0.9562
-0.9707
-0.7092
-0.2460
0.2872
0.7386
0.9801
0.9428
0.6375
0.1509
-0.3787
Prelude Rhythmicity.PolyRhythm Numeric> getPolyChRhData 'a' 5 (PolyCh [True,True,True,False] 5)
  (PolyRhythm [1,1,1,1,1]) . map (sin . (*27182.81828459045) . (*pi)) $  [0,0.01..0.24]
[[RP P a,RP P b,RP P e,RP P d,RP P c],[RP P d,RP P e,RP P c,RP P b,RP P a],[RP P a,RP P b,
  RP P c,RP P e,RP P d],[RP P d,RP P e,RP P c,RP P b,RP P a],[RP P a,RP P b,RP P c,RP P e,RP P d]]
```

Coherent States of Polyrhythmicity as One of the Essential Sources of Rhythmicity

The described pattern of rhythmicity is one of the significant possible options for the formation of rhythmicity in particular in lyrics or music, but not the only one. It should be noted that the described mechanism of rhythm formation, as is noticed in the statistical experiments with texts using this code (the code of the library and its dependent packages on the Hackage site) may not be the only possible option, but in many cases it is crucial and influences the course of the rhythmization process (formation, change or disappearance of the rhythm). It is also known that the presence of the statistical relationship does not mean the existence of deeper connections between phenomena, in particular – the causality. ”Correlation does not mean causality.” A deeper connection implies the presence of other than the statistical ones to confirm it.

Rap Music Consequences

The code of the library allows in practice to obtain rhythmic patterns that are often close to the lyrics in rap style. Therefore, this can be attributed to one of the direct applications of the library.[45]

A Child Learns to Read, or Somebody New to the Language

When a child just begins to read words in the language (or, there can be just somebody new to the language) he or she starts with phonemes pronunciation for every meaningful written (and, hence, read) symbol. Afterwards, after some practice, he / she starts to read smoothly. Nevertheless, if the text is actually a poetic piece, e. g. some poem, it is OFTEN (may be, usually, or sometimes, or occasionally, etc.) just evident that the text being read in such a manner has some rhythmicity properties, despite the fact that the phonemes are read and pronounced in a manner of irregular and to some extent irrelevant to the normal speech mode lengths (durations). We can distinguish (often) the poetic text from the non-poetic one just by some arrangement of the elements.

The same situation occurs when a person with an accent (probably, strong, or rather uncommon) reads a poetic text. Or in other situations. The library design works just as in these situations. It assumes predefined durations, but having several reasonable (sensible) ones we can evaluate (approximately, of course) the rhythmicity properties and some other ones, just as the algorithms provided here.

This, to the mind of the author, is a ground for using the library and its functionality in such cases.[44]

Increasing and Decreasing Functions

Since the 0.5.3.0 version of the phonetic-languages-rhythmicity package the increasing and decreasing functions for the polyrhythmicity evaluation have become more similar to be more likewise the inverse one to another. This leads to that fact that these functions now are expected to be smoother for usage for the beginning of the line, its middle and its ending.

Note: since the 0.6.0.0 version of the phonetic-languages-rhythmicity package the values of the properties from the series “c”, ”s”, “t”, “u”, “v” and the many others (starting from the 0.9.0.0 version) can be negative by sign. This does not influence the logics of the working library functions and programs. Since the 0.8.0.0 version of the package phonetic-languages-simplified-examples-array there were added also new properties that can be negative by sign.

Problem of choosing the best function and related issues

Consider the following question: suppose we have obtained the best version (in our subjective opinion or on the basis of some criteria, it is irrelevant here) of the line in one way or another (here the method does not really matter). Is there a function that makes this particular variant of the string optimal, i. e. for which such a variant of the string gives the maximum of all possible permutations? Yes, there is. This is easy to prove. The proof is reminiscent of the principle of equalizers.

Let $n \in N$ be the number of syllables in such a line. Arrange the durations of the syllables in ascending order (standard procedure for descriptive statistics). We will find the smallest nonzero difference between adjacent values, divide it by 5. Denote this value by δ. Now consider a number of syllable durations for our best string. Number each syllable from the beginning, counting from 1. Denote by $Y = \{y_i, \quad i \in N, \quad i = 1, 2, \ldots, n\}$ the set of all values of durations in the order of sequencing in the best line. Denote by $X = \{0 = x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_{n+1}, \quad i \in N, \quad i = 1, 2, \ldots, n\}$ the set of coordinates of the points of the ends of time conventional intervals, into which our best line divides the time line (the left edge is 0, because the countdown starts with 0). Denote by $M = \{z_i = \frac{x_i + x_{i+1}}{2}, \quad i \in N, \quad i = 1, 2, \ldots, n-1\}$ a set of midpoints of the segments into which the time line is divided by the conditional intervals ends. Denote by $L_1[a, b]$ the class of Lebesgue-integrable functions on the interval $[a, b]$. Let’s mark $I(y_i, z_i, \delta)[z_i - \delta, z_i + \delta]$ class of all bounded functions with $L_1[z_i - \delta, z_i + \delta]$, the maximum and minimum values of each of which lie on the segment $[y_i - \delta, y_i + \delta]$. We denote each function of class I by g.

Consider the class of functions F (a kind of finite analogues of the known delta functions of Dirac), defined as follows:

$$f(x,i) = \begin{cases} g \in I(y_i, z_i, \delta)[z_i - \delta, z_i + \delta], & \text{if } y_i \text{ is a unique value in the set } Y,\ x \in [z_i - \delta, z_i + \delta] \\ y_i, & \text{if } y_i \text{ has equal value with some other number from the set } Y,\ x \in [z_i - \delta, z_i + \delta] \\ 0, & \text{otherwise} \end{cases}$$

It is easy to see that

$$sum_{i=1}^{n} \int_{-\infty}^{\infty} f(x,i)\, \mathrm{d}x,$$

where integration is carried out according to Lebesgue, is the desired function (because only the syllables of the best string in their places, taken into account with their indices, give a positive contribution to its value, and for all other variants of the function at least some of the syllables give 0 contribution), and it is not unique due to the fact that in the first line of definition $f(x,i)$ the function can have an arbitrary value from a closed non-empty interval. Therefore, there is at least one class of functions that is described by such a formula, for each of which this particular variant of the string will be optimal.

Let's ask the following question: if we consider not a line, but their combination, for example, a poem. Does a function that makes each line optimal (i. e. which will describe the whole work, each line in it) exist for the whole work (for this whole set of lines)?

In this case, the previous method of constructing the function does not give the desired result, because for two lines it may be that what is best for one of them is not the best for the other. In the case of increasing the number of lines, this general suboptimality only intensifies. However, the existence of such a function for different particular cases is a fundamentally possible situation, however, the probability of its existence should decrease both with increasing number of rows and with the appearance of different features of rows, which increase the differences between them. In general, the search for just one such function may be virtually impractical.

Nevertheless, if we consider the whole work as one line, considering it optimal, then the method just described to construct the corresponding function (class of functions) again gives the result. Yes, the number of syllables in it (in the generalized "line"-work) increases, but the procedure itself gives similar results (if we neglect the possible identical durations and repetitions of syllables in a larger text, which lead to the fact that some words can be rearranged and that makes the text only approximately optimal). It is possible to suggest further improvement of this procedure (for example, introduction of the factors which depend on values of the next durations of syllables) that allows to reduce suboptimality of the text.

But still, if you go from one text to another, the resulting function is unlikely to be optimal.

We conclude that for the whole set of expediently organized texts the existence of a universal function is seen as a certain hypothesis (with almost certainly proposes a negative answer).

Connection with fractals

We will pay attention to this fundamental ”break”, the fundamental dissimilarity of the texts.

But first note the following. The function considered above makes one line variant optimal at the same time you can consider not one option, but several, from which to choose the best. Then the task is reduced to search optimal group of options with the minimum possible number of components.

If we consider again the boundary case of a group with the number of elements equal to the factorial of the number of words in a line, it is easy to see for combinatorial reasons that such a group contains (consists of, is) all variants of a string of these words. But for 5 words such a group contains $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$ options for 5 words, a total of 600 words to read which usually takes a few minutes.

If you reduce the number of words in the optimal group, then among them may not be the right option, but the reading time decreases accordingly. So, if we have some way of at least approximate ordering, it will be expedient to find the optimal ratio ”group size - the degree of accuracy of approximate ordering”.

This leads to the idea of using different functions that are easy to calculate, have a certain characteristic behavior and allow you to roughly organize the sets of all permutations according to the search task, and the search is no longer a single option, but the optimal group of options.

At the same time, the search stages are fundamental, when the program does not give the desired result completely. Then you can change something in the data (without changing the words themselves), or change the words to change the options that the program works with, because it changes the structure of the set of all options and allows you to get other still somewhat ”optimal” options from the author’s point of view and in terms of the program. This gave impetus to the so-called recursive mode of operation, when the change of data is the merging of words (and hence the exclusion from consideration of options where they are not in a row), as well as the mode of several options.

An interesting observation is also that in recursive mode you can get lines in which more noticeable are the ”breaks” of rhythm, which can often be included in the rhythm through the introduction of additional pauses. Poems with pauses (for example, so-called caesuras) are also known in theory, for which this may be useful.

Also the ability to form lines from lines, and each line from certain components, resembles fractals. However, the relationship with fractals requires more detailed and in-depth study.

An interested reader can turn to the literature.[24, 41, 19, 38, 49, 34, 31, 46, 39, 47, 40, 30, 29, 35, 37, 42, 33, 48, 32, 52, 51, 50, 36, 4, 1]

Analogy with the extremity principle

In mechanics and optics, the principle of extremity of action is known (a physical quantity of action is introduced, which for real processes in these areas takes among all possible values along the trajectories of the minimum (most often) or maximum value, ie in one word the extreme value). In thermodynamics there is a law of increasing entropy of closed systems. In the case of mechanics and optics, the search for real trajectories is reduced to finding those of the possible trajectories for which the value of the action functional is extreme, which allows you to use the apparatus of higher mathematics to find these extreme values.[28]

An analogy to this principle is the comparative mode of operation in combination with the mode of several properties.

Ability to use your own durations of representations of sounds or phonetic phenomena

The programs offer four different sets of phonetic representations by default but starting with version 0.13.0.0 it is possible to set your own durations. To do this, specify them as numbers of type 'Double' in the file in the order defined as follows:

```
UZ 'A' D       дз (plain)                8
UZ 'A' K       дз (palatalized)           9
UZ 'B' D       ж  (plain)                10
UZ 'B' K       ж  (semi-palatalized)      11
UZ 'C' S       й                     27
UZ 'D' N       сь                    54
UZ 'E' L       ч  (plain)                39
UZ 'E' M       ч  (semi-palatalized)      40
UZ 'F' L       ш  (plain)                41
UZ 'F' M       ш  (semi-palatalized)      42
    G                                55
    H          ю                     56
    I          я                     57
    J          є                     58
    K          ї                     59
```

```
    L           ’                       60
    M           ’                       61
    N           нт                      62
    O           ст                      63
    P           ть                      64
    Q           дзь                     12
    R           зь                      13
    S           нь                      65
    T           дь                      14
UZ ’a’ W        а                       1
UZ ’b’ D        б  (plain)                  15
UZ ’b’ K        б  (semi-palatalized)        16
UZ ’c’ D        ц  (plain)                  38
UZ ’d’ D        д  (plain)                  17
UZ ’d’ K        д  (palatalized)             18
UZ ’e’ W        е                       2
UZ ’f’ L        ф  (plain)                  43
UZ ’f’ M        ф  (semi-palatalized)        44
UZ ’g’ D        ґ  (plain)                  19
UZ ’g’ K        ґ  (semi-palatalized)        20
UZ ’h’ D        г  (plain)                  21
UZ ’h’ K        г  (semi-palatalized)        22
UZ ’i’ W        і                        6
UZ ’j’ D        дж (plain)                  23
UZ ’j’ K        дж (palatalized)             24
UZ ’k’ L        к  (plain)                  45
UZ ’k’ M        к  (semi-palatalized)        46
UZ ’l’ S        л  (plain)                  28
UZ ’l’ O        л  (palatalized)             29
```

```
UZ 'm' S      м  (plain)               30
UZ 'm' O      м  (semi-palatalized)     31
UZ 'n' S      н  (plain)               32
UZ 'n' O      н  (palatalized)          33
UZ 'o' W      о                         3
UZ 'p' L      п  (plain)               47
UZ 'p' M      п  (semi-palatalized)     48
UZ 'q' E      ь                         7
UZ 'r' S      р  (plain)               34
UZ 'r' O      р  (palatalized)          35
UZ 's' L      с  (plain)               49
UZ 't' L      т  (plain)               50
UZ 't' M      т  (palatalized)          51
UZ 'u' W      у                         4
UZ 'v' S      в  (plain)               36
UZ 'v' O      в  (semi-palatalized)     37
UZ 'w' N      ць                        66
UZ 'x' L      х  (plain)               52
UZ 'x' M      х  (semi-palatalized)     53
UZ 'y' W      и                         5
UZ 'z' D      з  (plain)               25
UZ 'z' K      з  (palatalized)          26
```

where the specified values in the list refer to the phonetic representations of the data type UZPP2 (from the module Languages.Phonetic.Ukrainian.Syllable.ArrInt8). The last column is 8-bit integers (GHC.Int.Int8), which represent these sounds in the new modules.

If you want to specify several such sets (up to 9 inclusive), you can specify '*' or several such characters from a new line, and then from the next line there will be a new set of values.

Each set should be in the following order: [-1,1,2,3,4,5,6,7,8,9,10,11,15,16,17,18,19,20,21,22,23,24,25,26, 27,28,29,30,31, 32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,66]

where the number corresponds to the last column in the above diagram. -1 corresponds to a pause between words (does not affect the search results of the line).

Then when executing the program somewhere among the command line arguments (it does not matter where exactly) specify «+d» <path to the file with the specified data>. Programs will read these values and convert them to the appropriate values. As properties it is necessary to use then those which begin with the letter «H», and further the corresponding designation of property merged with it. For example, «Hw04», the last digit in the record in this case will mean the ordinal number of the set of values, starting from 1 (maximum 9).

Along with the custom values, you can use the provided by the library ones, as usual, in the mode of several properties.

Minimum grammar for possible preservation of meaning and intelligibility

Programs use permutations of words that neglect any (or at least part of) grammatical connections, word order, and so on. This can lead (in addition to the need to think) to situations where grammatically related language constructions are broken, their parts are transferred to other places, forming new connections and changing the meaning of the text.

To reduce this, to eliminate some of these effects, programs use concatenation of words that have a close grammatical connection, so as not to break them in the analysis. This allows you to maintain greater semantic ease and recognizability of the text, as well as a side effect to increase the overall length of the line, which can be analyzed. In the Ukrainian language, grammatically related auxiliary or dependent words precede the independent or main one, so the concatenation of these auxiliary or dependent words to the next one is used. The completeness of the definition of such cases is not exhaustive, but the most frequent cases are considered.

To reduce this, to eliminate some of these effects, programs use concatenation of words that have a close grammatical connection, so as not to break them in the analysis. For the general case, it should be borne in mind that auxiliar or dependent words can go after the independent or the main, so this should be considered separately (and attach such words to the previous, not to the next). Currently, the generalized version of phonetic-languages-simplified-generalized-examples-array is implemented only similarly to the Ukrainian language version. However, generalization is not a technically difficult task and is planned in the future.

Pairwise permutations as a variant of the universal set of permutations

By default, the program analyzes the universal set of permutations of all words and their concatenations, while the number of analyzed options increases as a factorial of the number of such words or combinations. The text, organized more or less coherently in relation to one or another property, can be radically different from the original, which complicates understanding and has the effect of delaying calculations.

In order to quickly check the possible improvement of the text using the approach, only a pairwise permutation of two words or their combinations as a universal set is introduced. When executing programs, it is specified by the command line parameter «+p» somewhere among the arguments (position does not matter). In this case, only pairwise permutations are used. The number of words and their combinations that can be considered by the program as one line for analysis increases in this case to 10.

In this case, the analysis is much faster (because the number of cases is significantly reduced), and the text changes less, which allows you to keep it more recognizable. Consistent application of this case is possible, but it should be borne in mind that this may not be the best option in terms of the first approach, and it may take even longer than analyzing the whole set of permutations (because some options will be analyzed several times which does not happen in the first case). The following should be taken into account: when searching for the maximum element by the value of the property (ie without changing the structure) if the analysis of pairwise permutation received a text that coincides with the original, then there is a good chance that this option is optimal in terms of the property in question (although this is not guaranteed). And one more thing: in this case, a local maximum is reached (which may or may not be the global one). If the repeated application leads to the formation of another (already the third) option, then the previous local maximum was not exactly global and the program is moving in its direction.

See also: [3, 20, 22, 2, 25, 27, 21, 26, 10, 5, 11, 12, 17, 13, 14, 16, 15, 6, 7, 8, 9]

Prerequisites for using the software package

At the moment, the programs work for workstations (desktops e. g.), and there are no mobile versions.

You must have Haskell applications installed and configured:

1. GHC (versions not earlier than 7.10)

2. Cabal

The executables of these programs must be searchable through the PATH workspace variable (this is the default setting).

If possible, use the system package manager (programs) to install also important packages Haskell bytestring, vector, heaps, parallel. If you also plan to use r-glpk-phonetic-languages-ukrainian-durations, then also install and set the programming language (and better development environment) R.

If the required Haskell packages are not installed using the system manager, they will be installed when installing the packages (downloaded and installed automatically, with additional time also spent on their compilation).

Remark on terminology

Earlier versions of the packages used the names ‘norms’ and ‘metrices’ for the properties of the texts. Because in the sense typical of mathematics (including functional analysis) all these properties are not actually metrices and norms (e. g. the inequality of the triangle is not fulfilled), then the non-ambiguous ‘property’ will be used everywhere instead.

‘Property’ hereafter means the functional representation of the latter.

Installing the package

Open a command prompt or terminal and enter as commands:

```
cabal update
cabal --reinstall --force-reinstalls install phonetic-languages-plus
cabal --reinstall --force-reinstalls install phonetic-languages-simplified-examples-array
```

It is also recommended to install the following packages:

```
cabal --reinstall --force-reinstalls install r-glpk-phonetic-languages-ukrainian-durations
cabal --reinstall --force-reinstalls install mmsyn6ukr-array
```

The latter is optional, but useful for sound and does not take a lot of space.

If there are messages about outdated command variants, enter instead of update – v1-update, install –

```
--enable-split-sections --enable-split-objs --enable-library-stripping \
  --enable-executable-stripping v1-install
```

If this does not help, please, wait for the newer package versions that will have the issue being solved, or write to the maintainer to the email address olexandr543@yahoo.com..

In the newest version of cabal it is planned to use v2-* command versions by default, but there are some additional needed workarounds to implement different versions of the packages, libraries and executables, that is why it is recommended to use v1-* variants. For example, if after the installation in such a way (v2*) while loading a module into the interpreter GHCi there are error messages that the module is in the hidden package, then run the interpreter again with a flag -package <name of the module's package>.

The packages base, parallel and heaps can be often found in the OS repositories, the rest of them it is recommended to install from the Hackage server (using the above-mentioned commands).

Working with the program lineVariantsG3

Verify that the folder (directory) where cabal installed the executables of the programs is available for search in the PATH environment variable.

The program now supports two modes:

- with one property (normal mode);
- with several (not more than five different) properties.

The latter is used if the command line arguments include a group entered by delimiters +m <property type1> <numeric arguments1> <property type2> <numeric arguments2> <property type3> <numeric arguments3> <property type4> <numeric arguments4> <property type5> <numeric arguments5> -m.

More about this at relevant section (see link above). The operation of the program in this mode is described later in a separate section.

To work in the single property mode, enter the command at the command prompt (or terminal):

```
lineVariantsG3 <the first argument> [<WX argument> <whether to print property value(s)>
  <whether to fix the last word> <stay in place> <whether to use recursive interactive mode>]
    [<whether to use the multiple sources mode>] <numerical arguments> < property type>
      <Ukrainian text>
```

everything here and further in the single line or using the terminal line hyphenation, or:

```
lineVariantsG3 <the first argument> [<WX argument> <whether to print property value(s)>
```

```
  <whether to fix the last word> <stay in place > <whether to use recursive
    interactive mode>] <numeric arguments > <property type >
      [<whether to use the multiple sources mode>] <Ukrainian text>
        <somewhere in the middle arguments as a single group: restrictions>
```

and press Enter. Additionally, you can set the interactive mode, about what see below for more details .

If you don't specify groups in square brackets, you'll see something like this:

```
lineVariantsG3 10.0_1.2 yy2 садок вишневий коло хати хрущі над вишнями гудуть
```

(Ukrainian text entered in the end of the command line arguments - an excerpt from a famous poem by Taras Shevchenko; in general this line is the entered command). Here and further is cited by: [23]

```
садок колохати хрущі гудуть надвишнями вишневий
```

(in general, there may be several such variants that form one group, as well as several such groups; all groups follow one another from top to bottom in descending order of the final value of the property, which maximizes the selected property for given intervals)

```
[3.6562] (value of selected property before applying interval conversion)
[3.6562] (the value of the selected property after the interval conversion, the final
         value of the property for this line).
```

Note that the text may (and probably will not) be written the way it is spelled according to spelling and punctuation rules, but you can read it and try to understand. By modifying the first arguments entered, you will (most likely) get other output, the same obviously, it also applies to the Ukrainian text. Too long text will be reduced to a volume that you could understand (perhaps after previously mentioned training) without effort.

Try to evaluate by reading the variant how it is suitable.

NOTE: Also keep in mind that in single property mode, numeric arguments precede the notation of a property, and in the case of multiple properties (see below) on the contrary - the designation of the property begins a set of numerical arguments that relate to it, if any (otherwise default values are used, which are just the same to the search for the maximum element).

Ukrainian information messages

In order for the program to display informational messages in Ukrainian (it displays in English by default), it is required to specify ‘+u’ as one of the command line arguments somewhere outside the option groups, for example, at the beginning. You can also customize the alias for this version of the program, as for other options, if you like. Refer to the documentation for details command shells.

More complex usage

Numeric arguments, if specified, have the following meaning.

The first numeric argument is the number of groups with the same maximum property value (in descending order) that will be output on the screen as a result. If you specify a larger number than there is at all, then all possible results are displayed that satisfy all other conditions. If no numeric arguments are specified, it is considered equal to 1.

The second numeric argument is the number of intervals into which the interval between the minimum and maximum value of the property for this line. If not specified, it is considered equal to 1. A value of 0 does not allow other numeric arguments to further change the result of the work of the program.

All subsequent numeric arguments (if specified, otherwise no permutations occur) are interval numbers that will be swapped with the maximum number. This allows you to change the structure of the data that is displayed as a result of the program and see the internal (not maximum) items. For example, the numeric arguments 2 6 1 4 (in this order) will mean that during execution the program will return 2 groups of elements with the maximum values of the property (the largest and the next largest ones) obtained after permutations of the intervals; the interval between the maximum and minimum value of the property will be divided into 6 equal intervals, thus elements that are in the first and fourth, counting from the minimum (interval number 1) will be moved to the largest one; the command will display 2 groups of elements. Values that were in the maximum interval will be moved to the interval with the lowest number among those that are moved to the maximum one. Thus, at an output these values will be deduced at the latest.

Parameter +l (+bl) and its usage

NOTE: If there were no +l, +bl, +i, +y ... command line arguments, then each output block will have 2 numbers in square brackets displayed - the initial value of the property (without moving the intervals) and the value after moving. If there was (at least) one of these (groups) of characters - the value of the properties will not be printed.

It should also be remembered that:

```
+bl == +b +l
```

(this is just a reduction in the use of both parameters at once, instead of 5 characters you only need to enter 3).

If you also specify +f ... or +i, then this parameter may not be specified (it will be applied automatically), instead, if you want, you can specify an additional +b instead.

Parameter +b (+bl) and its usage

If somewhere among the arguments of the command line specify an argument in the form of +b (or +bl), the program will preserve, when outputting and analyzing, the last word in the line in its place - it is very convenient when you need, having a rhyme, to pick up other words. If not specified, then all words will be moved (if necessary). The operation of the parameter is actually implemented as an additional constraint , see the following sections. You can also set additional constraints.

```
+bl == +b +l
```

(this is just a reduction in the use of both parameters at once, instead of 5 characters you only need to enter 3).

About the use of other parameters a little later.

Multiple properties mode (+m …-m)

If you specify a group of arguments selected by the +m and -m delimiters from the command line arguments so that the argument group is selected +a and -a delimiters were not inside this, and vice versa (so that they do not intersect), then the program will work in the multiple

properties mode. The values of the properties will not be displayed on the screen, instead it is possible to set no more than four different properties and to each of them to specify arguments (see: More complex usage). The program will then find variants that meet each of these conditions, and then display only those variants that are found in all selected and given properties with parameters. Numerical arguments that stand after the property designation and precede the next denote a property related to that property. If numeric arguments are omitted, the default values are used (in fact, this is equivalent to simply searching for maximum property values).

In general, this is the more comprehensive use of this program.

Try, for example, to specify:

```
lineVariantsG3 +m 02y 3 03y 3 y0 10 -m +bl <Ukrainian text>.
```

Interactive mode (+i) and its usage

Interactive mode (additional extended user interaction, in addition to the required) is enabled and set accordingly by the command line argument +i, which can be placed anywhere in the command line. In this case, the program displays not just lines that satisfy all conditions, but for each line also displays its sequence number (starting with 1) in order of increasing the ‘weakness’ of all conditions (the higher the number in the general case, the more likely the weaker effect is of given conditions, although this is not always the case - in particular when you need to withdraw only one group). The program then asks what the choice is waits for the option number entered by the user. Then returns that option without a number.

It looks something like this:

```
lineVariantsG3 +i +m 02y 10 0y 10 y0 50 y2 40 -m садок вишневий коло хати хрущі над
  вишнями гудуть
Please, check whether the line below corresponds and is consistent with the
constraints you have specified between the +a and -a options. Check also whether
you have specified the ”+b” or ”+bl” option(s). If it is inconsistent then enter
further ”n”, press Enter and then run the program again with better arguments. If
the line is consistent with your input between +a and -a then just press Enter
to proceed further.
садок вишневий колохати хрущі надвишнями гудуть
```

```
1 вишневий колохати хрущі надвишнями садок гудуть
2 вишневий колохати садок хрущі надвишнями гудуть
3 вишневий колохати хрущі садок надвишнями гудуть
4 вишневий садок колохати хрущі надвишнями гудуть
5 колохати вишневий надвишнями садок хрущі гудуть
6 хрущі садок вишневий колохати надвишнями гудуть
7 колохати хрущі садок вишневий надвишнями гудуть
8 хрущі колохати садок вишневий надвишнями гудуть
9 хрущі садок колохати вишневий надвишнями гудуть
10 хрущі надвишнями садок вишневий колохати гудуть
11 колохати хрущі надвишнями садок вишневий гудуть
12 надвишнями колохати хрущі садок вишневий гудуть
13 надвишнями садок колохати хрущі вишневий гудуть

Please, specify the variant which you would like to become the resulting string by its
  number.
4
вишневий садок колохати хрущі надвишнями гудуть
```

Interactive mode of writing a line to a file (+f ...)

If you specify a group of three arguments as +f <path to record file>, then in the specified path to the text file specified, if possible, the final result of the program will be appended, except that it will be displayed as in the usual interactive mode on the screen. This command line arguments group can be anywhere between the command line arguments of the program call, but should not be contained inside other arguments of the form +a ... -a, +m ... -m, etc.

The result can be something like this:

```
lineVariantsG3 +f hello.txt +bl +m 02y 10 0y 10 y0 50 y2 40 -m садок вишневий коло хати
```

```
  хрущі над вишнями гудуть
Please, check whether the line below corresponds and is consistent with the constraints
you have specified between the +a and -a options. Check also whether you have specified
the ”+b” or ”+bl” option(s). If it is inconsistent then enter further ”n”, press Enter
and then run the program again with better arguments. If the line is consistent with
your input between +a and -a then just press Enter to proceed further.
садок вишневий колохати хрущі надвишнями гудуть

1 вишневий колохати хрущі надвишнями садок гудуть
2 вишневий колохати садок хрущі надвишнями гудуть
3 вишневий колохати хрущі садок надвишнями гудуть
4 вишневий садок колохати хрущі надвишнями гудуть
5 вишневий садок хрущі надвишнями колохати гудуть
6 хрущі садок вишневий колохати надвишнями гудуть
7 садок хрущі вишневий надвишнями колохати гудуть
8 колохати хрущі садок вишневий надвишнями гудуть
9 надвишнями колохати хрущі вишневий садок гудуть
10 хрущі колохати садок вишневий надвишнями гудуть
11 хрущі садок колохати вишневий надвишнями гудуть
12 хрущі надвишнями садок вишневий колохати гудуть
13 колохати надвишнями садок хрущі вишневий гудуть
14 колохати хрущі надвишнями садок вишневий гудуть
15 колохати хрущі садок надвишнями вишневий гудуть
16 надвишнями колохати садок хрущі вишневий гудуть
17 надвишнями колохати хрущі садок вишневий гудуть
18 хрущі колохати надвишнями садок вишневий гудуть
19 хрущі колохати садок надвишнями вишневий гудуть
20 надвишнями садок колохати хрущі вишневий гудуть
21 садок надвишнями колохати хрущі вишневий гудуть
```

```
22 хрущі надвишнями колохати садок вишневий гудуть
23 надвишнями садок хрущі колохати вишневий гудуть
24 хрущі надвишнями садок колохати вишневий гудуть
25 садок хрущі надвишнями колохати вишневий гудуть

Please, specify the variant which you would like to become the resulting
  string by its number.
4
вишневий садок колохати хрущі надвишнями гудуть
```

This end line in the program output will also be appended to the file with the specified name, if possible for this user.

If you wish, you can run the command again with new text and / or new arguments. If similarly the same file is specified, then the new result will be appended further to the same file. This makes it possible to apply this program consistently, write or rewrite texts (e.g. poems).

Mode of simultaneous possible variations of the text

Starting with version 0.3.0.0, the ability to process several variations of text at once has been added, in particular those that deals with synonyms, paraphrases, etc.

To do this, use the following special construction instead of plain text as extreme arguments:

```
{<variant1 of the Ukrainian text> / <variant2 of the Ukrainian text> / ... /
  <variantN of the Ukrainian text>}
```

everything at the single line with at least two variants inside curly braces. These options will be worked out in turn each in particular during one call of the program, and you will select one of the variants (possibly the empty one). In the end there will be an opportunity to choose among these pre-prepared versions of the only one, single final variant, which will be the result (and accordingly, for example, will be displayed and written to a file if provided by command line arguments).

Please note that the program in this mode provides processing of each of the possible combinations of variations, and therefore, if you specify too many of them (for example, 3 variations on one word and 4 on another will create 3 * 4 = 12 variations which will be consecutively processed) the execution time of the program can be longer than expected until you get the final result.

Recursive mode of the work (“+r”)

Starting from the version 0.9.0.0 you can execute the program in the recursive interactive mode. For this, you need to run the command with the parameter “+r”, e. g. at the beginning after the first argument. In such a case the program will be executed recursively, proposing to end the recursion on every step. The result of the last step will be the overall result of the program execution.

This mode is incompatible with constraints (because the constraints do not have the proper expected meaning and begin to ‘shift’ from the needed parts of the text to the other ones), that is why they should not be used simultaneously, this mode can be the alternative to the latter ones. While execution in the recursive mode there is no possibility to change the call parameters for the properties etc., therefore, choose them wisely.

The text changes in this mode are specified by the so called ‘interpreter string’, i. e. the textual input that in the arithmetic expression – either in a number or division expression encodes the following actions of the program.

- If on the interpreter string the two-digit number is entered then the first digit the program tries to interpret as an index of the first word (starting counting from 1) to which the change is applied, the second digit is interpreted as a number of the words that are concatenated further including the first specified one. Afterwards, the program (if it is possible to do so without an error) works with the newly generated text. For example, the “12” means that the program will concatenate the first word (digit “1”) with the following ones in the quantity of 2 together (digit “2”) this leads to concatenation of the first two words. “34” means that the program tries to concatenate consequently 4 words starting from the third one. If it is impossible the program will execute the previous stage again with prompt to input the interpreter string again.

- If the three- or multi-digit number is specified then all the digits that are not equal to 0 the program tries to interpret as the indices of the words that are needed to be concatenated in the order of the digits written down in the interpreter string.

- If the digit (greater than 0 and not greater than the number of words on the line) precedes the ‘/’ and then is followed by integer number with sign. Then the first digit before the ‘/’ sign means the index of the words (counting starts from 1) that is splitted into 2 parts, the second number is the quantity of the symbols that the program counts from the beginning of the word (in case of positive integer) to the right or from the end of the word (in case of negative integer) to the left to split it into two parts (the positive case is similar to the work of the standard function ‘splitAt’). Then the program, if the specified string was successfully interpreted in such a way, will split the specified word into two parts (one of which can be an empty string) and will work with the text in which the specified word is substituted with these two new words (or just the first of them, if they are not empty and the number of the words in the line

is already equal to 7 at the moment of splitting). For example, the interpreter string “1/5” will split the “садоквишневий” if it is the first word in the text into “садок вишневий” (counting from the beginning the 5 symbols) and will work with the new text further.

WX argument

If among properties you use “w” or “x” series (or both ones) then you can specify for them another argument that must begin with “+x” with the next written down two positive Double numbers connected by the ‘_’ symbol (underline symbol). For example, +x2.345_0.45676237876. If this argument is not specified explicitly then the default one +x2.0_0.125 is used.

The first number is used as a factor (multiplier or divisor in case of non-suitability) and it mostly influences the property value, it deals with the most important syllables in the rhythmic group; on the other hand, the second one is used either just for increasing of the property value if the less highlighted syllable durations in the rhythmic group (corresponds to ‘w’ series) or also for more complex behaviour (‘x’ series).

More details can be found in the section Types of properties.

Multiple sources usage mode

If among the command line arguments before the Ukrainian text (or instead of one) or after it one specifies a group of aguments in the

```
*t_ ... ^t
```

frame then the program will work using the multiple sources mode. Instead of underline character here and ellipsis you can specify also arguments.

The program’s behaviour differs for the cases of pairwise permutations (see: Pairwise permutations as a variant of the universal set of permutations, it is set using the command line argument «+p») and the full universal set of permutations.

In the first case, at the place of underline the program waits for one of the numbers of: 10, 11, 20, 21, 30, 31, 40, 41, 50, 51, 60, 70, 71, 80, 81, 90, 91, where 0 as the second digit means that the program won’t have concatenate the lines from each source into one singular line before its analysis and processments, and the 1 there means that the program will at first concatenate all the lines of each source into the one single line and only then it will apply the splitting into the parts and other processments specified by other arguments. The

first digit (if not equal to 1) means the number of the words in the lines (except the last one in every source) that the program will analyse consequently. If the first digit equals to 1, then this is equivalent to the 10 (and this is the maximum number of the words in this case).

In the second case, on the underscore sign the program expects one of the following numbers: 20, 21, 30, 31, 40, 41, 50, 51, 60, 61, 70, 71. Their meaning is the same as for the first case.

At the place of the ellipsis you can specify the paths to the files with Ukrainian texts which is used for the analysis. If they are left empty then the program will prompt every time the new line in the non-terminating loop (interruptible by the user) and, afterwards, will process it accordingly to all the other arguments. You can also in such a case just specify *t (without any other arguments).

Just try, e. g.:

```
lineVariantsG3 +r 3 w04 *t71 ”sadok.txt” ”other_poem.txt” ”just_text.txt” ^t
lineVariantsG3 +r 3 w04 *t71
lineVariantsG3 +r 3 w04 +f ”fileForSaving.txt” *t51
```

where it is expected that the specified files do exist and can be read (the program ignores the files it cannot read).

This mode is also incompatible with the constraints (+a ... -a), instead you should use the recursive mode.

Work with program unconcatUkr

While execution of the lineVariantsG3, rewritePoemG3 there become present some word concatenations for the preserving the minimae grammar and better text understanding. Some of them are typical and they can be easily distinguish from the usual (not concatenated) words. Therefore, for the speeding up the text post-editing with the programs usage you can use also the other program unconcatUkr.

```
unconcatUkr 1 <path to the file with the concatenated words> [<path to the new file>]
unconcatUkr 2 <path to the file with the concatenated words> [<path to the new file>]
unconcatUkr -i
```

are the three principal ways to utilize it.

The first one is the ‘safest’, it uses just those unconcatenations, which almost for sure won’t lead to the mistakes in the words. The second one is more ‘risky’ (aggressive) but has more effect, nevertheless, use it with caution and be aware of its possible consequences (some not so widely spread words will be wrongly splitted while the execution).

By default, if the number is not specified, then it is equal to 1.

Working with propertiesTextG3 (and distributionTextG)

Option I (lines only)

Verify that the folder (directory) where cabal installed the executables of the programs is available for search in the PATH environment variable.

Then enter the command at the command prompt (or terminal):

```
propertiesTextG3 <the first argument> [<WX argument>] [<whether to use
  ‘growing lines’>] <file with Ukrainian text> <control of the number
    of intervals> <control whether to print also text string> <control the
      division of text into lines> <property type>
```

and press Enter.

You will see something like:

```
propertiesTextG3 2.1_3.0 ~/sadok.txt s 1 0 04z
5
2 2 11 1.0000 5.6200 5.6200 0.30211480 2 1 Тарас ШЕВЧЕНКО
Вказематі
81 83 227 1.0222 2.7902 2.7297 0.53936831 3 1 Садок вишневий колохати
4 10 10 2.2936 2.2936 1.0000 1.39275766 3 5 Хрущі надвишнями гудуть
```

```
0 4 7 16.4444 26.2400 1.5957 1.20737478 3 4 Плугатарі зплугами йдуть
36 81 82 2.2345 2.2621 1.0123 1.36997886 3 5 Співають ідучи дівчата
4 4 16 1.1111 4.0625 3.6562 0.43895748 3 1 Аматері вечерять ждуть
81 144 146 1.7778 1.7942 1.0092 1.27248256 3 5 Сем’я вечеря колохати
5 9 16 1.8000 3.2000 1.7778 0.85714286 3 2 Вечірня зіронька встає
1 9 9 8.7059 8.7059 1.0000 1.79393939 3 5 Дочка вечерять подає
36 146 147 4.0625 4.0711 1.0021 1.60221297 3 5 Амати хоче научати
0 4 14 9.0000 32.0400 3.5600 0.54479419 3 2 Так соловейко недає
81 82 145 1.0014 1.7804 1.7778 0.72037537 3 1 Поклала мати колохати
4 4 10 1.0625 2.3906 2.2500 0.62672811 3 1 Маленьких діточок своїх
4 5 16 1.2162 3.9527 3.2500 0.49113233 3 1 Сама заснула колойіх
36 36 91 1.0038 2.5266 2.5170 0.56929225 4 1 Затихло все тілько дівчата
0 0 4 1.0000 9.0000 9.0000 0.20000000 2 1 Тасоловейко незатих
Міжітравня
С-Петербург
```

Numeric columns have the same value for both options. The difference is that in the second case the statistics on the whole text is more important from the point of view of the researcher than for each line in particular.

Column I is the minimum possible value of the selected property for the given data among all possible variants of permutations of words in line;

Column II is the actual value of the selected property for the specified data in the row, the one that is implemented in this particular version of the row;

Column III is the maximum possible value of the selected property for the given data among all possible variants of word permutations in line;

Column IV is the ratio of the value of the property for a given row and its minimum value for the words that make up the row; a numeric, which is not less than 1.0;

Column V is the ratio of the maximum value of the property for the words of this line and its minimum value, which consists of line; a number that is not less than 1.0 and not less than the number in column IV;

Column VI is the ratio of the maximum value of a property for a given row and its actual value; a number that is not less than 1.0 and not greater than the value in the V column;

Column VII is the ratio of the actual value of the property to the arithmetic mean (half-sum) of the maximum and minimum values for all possible permutations of words for a given data; a number that is displayed with the full calculated number of characters after dots; is important for further statistics for the whole text;

Column VIII is the number of words in a line, some of which may consist of several connected Ukrainian words;

Column IX is the number of the interval (starting with 1), which includes the actual value of the property for the specified data;

Further to the right - if <control whether to print also a line of text> as ‘1’, then a line of text that is being analyzed is displayed here; otherwise, it is not displayed.

Option II - statistics throughout the text (+ possibly lines)

Verify that the folder (directory) where cabal installed the executables of the programs is available for search in the PATH environment variable.

Then enter the command in the terminal:

```
propertiesTextG3 <the first argument> [<WX argument>] [<whether to use
  ‘growing lines’>] <file with Ukrainian text> <control of the number of
    intervals> <control whether to print also text string> <control the
      division of text into lines > <property type> | distributionTextG
        <same row argument> <whether also display ordinal data>
```

and press Enter. In Unix-like operating systems, the vertical line (highlighted in red) is used to create pipelines in the shell terminal; for Windows:

```
PowerShell -Command ”propertiesTextG3 <the first argument> [<WX argument>]
  [<whether to use ‘growing lines’>] <file with Ukrainian text> <quantity
    control intervals> <control whether to print also text string> <control
      the division of text into lines> <property type> | distributionTextG
        < same row argument > <or also display ordinal data >’
```

You will see something like:

36

```
propertiesTextG3 2.1_3.0 ~/sadok.txt s 1 0 02y | distributionTextG s 1 +W
        4
3       3       5       1.0000  1.7000  1.7000  0.74074074   2     1     Тарас ШЕВЧЕНКО
        Вказематі
108     108     142     1.0000  1.3111  1.3111  0.86538462   3     1     Садок вишневий колохати
4       34      34      8.5180  8.5180  1.0000  1.78987107   3     4     Хрущі надвишнями гудуть
3       25      25      8.1680  8.1680  1.0000  1.78184991   3     4     Плугатарі зплугами йдуть
108     109     110     1.0049  1.0194  1.0145  0.99518569   3     2     Співають ідучи дівчата
4       4       34      1.0000  8.5180  8.5180  0.21012893   3     1     Аматері вечерять ждуть
108     245     245     2.2694  2.2694  1.0000  1.38827528   3     4     Сем’я вечеря колохати
3       34      34      11.1680 11.1680 1.0000  1.83563445   3     4     Вечірня зіронька встає
3       25      89      8.1680  29.6720 3.6327  0.53260303   3     1     Дочка вечерять подає
108     277     277     2.5611  2.5611  1.0000  1.43837754   3     4     Амати хоче научати
3       3       13      1.0000  4.3111  4.3111  0.37656904   3     1     Так соловейко недає
108     142     245     1.3111  2.2694  1.7309  0.80203908   3     1     Поклала мати колохати
3       3       25      1.0000  8.1680  8.1680  0.21815009   3     1     Маленьких діточок своїх
3       12      25      4.0000  8.1680  2.0420  0.87260035   3     2     Сама заснула колойіх
27      29      242     1.0778  8.9778  8.3299  0.21603563   4     1     Затихло все тілько дівчата
4       4       12      1.0000  3.0508  3.0508  0.49372385   2     1     Тасоловейко незатих
        Міжітравня
        С-Петербург
--------------------------------------------------------------------------------
1       2       3       4
9       2       0       5
56.25%  12.50%  0.00%   31.25%
0.9098+-0.5590  0       16
2       3       4       5       6       7
2       13      1       0       0       0
********************************************************************************
```

```
2      .      .      .
6      2      .      5
1      .      .      .
.      .      .      .
.      .      .      .
.      .      .      .
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
2      0      0      0
6      2      0      5
1      0      0      0
0      0      0      0
0      0      0      0
0      0      0      0
```

There will be no color highlighting, just different types of statistics are grouped here (see the table below for semantics).

1. Red colour – Decimal fraction with error – The arithmetic mean (mathematical expectation) of all numbers in column VII statistics for rows, plus or minus standard quadratic deviation; in the case of the selected properties ”y0” - often a number close to 1.0; may contain a rounding error caused by finding the sum of numbers with floating point. In case all lines of text excluded from the analysis (see: explanation for orange) only one line of text - signal inscription is displayed instead of matrices and further information:

   ”1,000+-0,000!”,

   where ”⌢” means a tab character. This label means that the specified text is not suitable for program analysis as well as that any data that is gotten during the execution of the program (could be derived from this text and related texts) should be analyzed as follows, so as not to refute the conclusions made on based on the whole set of texts. Simlply speaking, in this case, you can not take into account the text, because with the right approach to the analysis and interpretation of data it should not break the results.

2. Blue colour – Natural numbers – Row distribution matrix by number of words and intervals; matrix element in the k-th row and j-th column - the number of rows for which the value of the selected property with the specified data falls in the interval with the number j (numbering starts with 1), which is equal to matrix column number, and contains in the line k words (words or combinations thereof that are displayed merged to comply with minimum grammar rules in data analysis and output), the number k is in the range between

2 and 7 inclusive (rows that are analyzed for the matrix contain from 2 to 7 words (or written together their concatenations)). Thus, the matrix always has 6 rows, and the number of columns will depend on entered and available data. In fact, this matrix replaces the graphical two-dimensional data distribution. It is displayed twice, one after the other delimited by tildes. In the first case zero values are not displayed, instead there are dots. This is an element of data visualization, which allows you to better 'see' how a distribution looks like where numeric values correspond to the 'height' on the distribution graph (the value of the discrete two-dimensional function distribution). In the second case in place of dots there are the corresponding values, which are all equal to 0. Data for the matrix are obtained from VIII and IX columns of statistics by rows.

3. Orange colour – Non-negative integers – Total number of rows. The first number on the left is the number of rows that are excluded from the analysis for the matrix, because they have little data (1 or less words). Equality 0 means that all lines displayed on the screen participate in the analysis for formation of a distribution matrix. Number on the right - the total number of lines in the text that are displayed and analyzed (including those rows that are listed on the left).

4. Green colour – Percents – Distribution of the total number of rows by intervals. The sum of percentage values naturally equal to 100%. Interval numbers inscribed above the corresponding percentage values. For example, the inscription in these three lines type:

   1 2

   10 15

   40% 60%

   means that of the total number of rows, which can be analyzed using the program (contain enough data), 40% is accounted for by the first interval (with a smaller value properties), and 60% - by the second one (according to greater value of the property). That is such lines 10 and 15, respectively. All intervals are equal in size, but can have different numbers of rows. This is a simple one-dimensional distribution, it is possible to build a histogram.

5. Yellow colour – Natural numbers – Interval numbers. The countdown begins with 1. Further below they correspond to the number of lines, values of the properties for which according to the data falls into the corresponding number interval.

6. Brown colour – Natural numbers – Number of words in lines. Lies within 2 up to 7 inclusive (if there are less words, then the line gets a value of 1.0 and is removed from analysis program for the matrix). Under them - the corresponding values of the number of such lines. 0 corresponds to the case of the absence of lines with the number of words (or concatenations that are displayed as one word).

Careful study of these data allows us to draw certain conclusions about the text, their totality, the model and language itself.

Statistics mode by multiple properties (+m ... -m)

Now, as for the program lineVariantsG3, you can use the mode of multiple properties. To do this, instead of one property it is possible to specify multiple ones in the block selected by delimiters +m ... -m.

In this case, the program will display something like:

```
propertiesTextG3 sadok.txt s 1 0 +m y0 0y 02y 03y y2 y3 yyy -m
2       4       2       1       2       4       4       1       Тарас ШЕВЧЕНКО
                                                                Вказематі
3       4       4       1       2       4       2       1       Садок вишневий колохати
3       4       1       4       4       4       4       4       Хрущі надвишнями гудуть
3       4       4       4       1       4       1       1       Плугатарі зплугами йдуть
3       2       4       1       1       2       1       4       Співають ідучи дівчата
3       2       2       1       1       1       1       4       Аматері вечерять ждуть
3       1       3       4       1       3       1       3       Сем’я вечеря колохати
3       3       4       4       4       4       4       2       Вечірня зіронька встає
3       4       2       1       4       2       4       1       Дочка вечерять подає
3       4       4       4       1       4       2       1       Амати хоче научати
3       4       1       1       4       1       4       4       Так соловейко недає
3       4       4       2       1       2       4       2       Поклала мати колохати
3       1       4       1       1       1       1       4       Маленьких діточок своїх
3       1       4       2       4       1       2       4       Сама заснула колойіх
4       3       3       1       1       1       1       1       Затихло все тілько дівчата
2       4       4       1       1       1       1       4       Тасоловейко незатих
                                                                Міжітравня
                                                                С-Петербург
```

In this case, you do not need to use the distributionTextG program, because its behaviour is not defined here.

The first column (highlighted in color here) - the number of words in the appropriate rows; the columns follow in the order in which they appear marked in the block of several properties, respectively - the numbers of the intervals, which include the values of the corresponding properties. The first number on the right (highlighted in red and the only one in its line) is the number of intervals for each property (they are all the same). Eight columns in this case to text records - means that there were 7 (= 8 - 1) given properties in the block.

White String Mode

The program can also now use ‘white lines’ mode, which means that lines that contain fewer words than needed to ensure the existence of at least two variants of the string, do not display statistics and it is not included in the overall result. Then in the case of one metric and the use of the distributionTextG program, you need to call the latter with an additional one argument +W (means whitelines).

For example, in this case you will see:

```
propertiesTextG3 sadok0.txt s 1 0 03y +b | distributionTextG s 1 +W
        4
        Тарас ШЕВЧЕНКО
        Вказематі
52   52   81   1.0000  1.5577  1.5577  0.78195489    3     1     Садок вишневий колохати
4    14   14   3.2040  3.2040  1.0000  1.52426261    3     4     Хрущі надвишнями гудуть
1    1    4    1.0000  2.7692  2.7692  0.53061224    3     1     Плугатарі зплугами йдуть
36   36   40   1.0000  1.1111  1.1111  0.94736842    3     1     Співають ідучи дівчата
1    1    1    1.4444  1.4444  1.0000  1.18181818    3     4     Аматері вечерять ждуть
36   36   52   1.0000  1.4444  1.4444  0.81818182    3     1     Сем’я вечеря колохати
14   14   14   1.0000  1.0000  1.0000  1.00000000    3     2     Вечірня зіронька встає
1    14   14   14.2400 14.2400 1.0000  1.86876640    3     4     Дочка вечерять подає
37   37   37   1.0000  1.0000  1.0000  1.00000000    3     2     Амати хоче научати
1    4    4    4.0000  4.0000  1.0000  1.60000000    3     4     Так соловейко недає
36   40   40   1.1111  1.1111  1.0000  1.05263158    3     4     Поклала мати колохати
1    1    1    1.0000  1.0000  1.0000  1.00000000    3     2     Маленьких діточок своїх
1    4    4    4.4444  4.4444  1.0000  1.63265306    3     4     Сама заснула колойіх
```

```
10   10   37   1.0000  3.7000  3.7000  0.42553191    4     1     Затихло все тілько дівчата
        Тасоловейко незатих
        Міжітравня
        С-Петербург
-------------------------------------------------------------------------------------------------
1       2       3       4
5       3       0       6
35.71%  21.43%  0.00%   42.86%
1.0974+-0.4077  0       14
2       3       4       5       6       7
0       13      1       0       0       0
*************************************************************************************************
.       .       .       .
4       3       .       6
1       .       .       .
.       .       .       .
.       .       .       .
.       .       .       .
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
0       0       0       0
4       3       0       6
1       0       0       0
0       0       0       0
0       0       0       0
0       0       0       0
```

The ‘white’ lines here are shown as some space where there would not be it otherwise.

Pairwise Permutations Mode

In the pairwise permutations mode the both programs propertiesTextG3 and distributionTextG can work. In such a case as an command line argument must be specified «+p» for both of them. In such a case, the statistics is displayed so as there are the possible number of the words in the line between 2 and 10 inclusively.

Fixed-line statistics (+b)

If you specify as one of the command line arguments for propertiesTextG3 characters +b, the program will calculate all statistics, as if the last word is fixed by constraint and does not move. In fact, in this case, the meaning of this symbol (argument) is similar to the lineVariantsG3 program. It is necessary to remember that it narrows a range of admissible values of properties and at invariable quantity of lines changes distributions inside the intervals.

Control the number of intervals

There are three possible cases:

- “s” – the number of intervals will be determined by the well-known Sturge’s rule where the number of tests will be equal to the resulting number of lines;
- “l” – the number of intervals will be determined on the recommendation of V. P. Levinsky (see: Опря А. Т. Статистика (модульний варіант з програмованою формою контролю знань). – Навч. посіб. – К.: Центр учбової літератури, 2012. – 448 с. ISBN 978-611-01-0266-7. С. 60);
- the number of intervals will be a natural number (must be greater than 1, although this is not checked);
- something else - will be used 9.

You can also control whether to print lines of the text

If this argument is 1, then to the right of the numerical data of the ordinal statistics a line that is analyzed will also be displayed (already converted for analysis). Otherwise the string will not be displayed in the output.

Control the division of text into lines

If you specify 1 here, the text will firstly be grouped into one line, and then divided into lines by the method of division in half (by the number of words or their cocatenations) until the length of all lines is less than 8 words or their combinations. Foreign characters will be filtered.

If set to 0, the text will be parsed after filtering out extraneous characters (approximately) in the lines that were originally.

Whether to use ‘growing lines’ mode

If among the command line arguments you specify “+gab”, where a, b are some digits except 0, then the ‘growing lines’ mode is used. This means that the text is transformed in a way so that at first the maximum number of words in lines is no greater than the second digit and then the lines are grouped so that the number of words in every line is close to the first digit if it can be achieved by concatenation of the lines into the single one consequently. In other words, there will be done some transformation of the text by lines to make the number of words in every line closer to the first digit and no more than 7 (for the last one the quotient is used). In such a case, the control of division of text into lines plays less role, but it can influence if it is equal to 1. For example, “+g73” as a command line argument means that after the application of the division of the lines all them are partitioned so that in every line there are at first no more than 3 words or their concatenations and then the lines are concatenated so that in every of them there are a number of the words close to 7 (and no more).

The same argument for the number of rows

It means there should be the same value, as in the place of control of the number of intervals.

Whether to display serial data as well

Here it is necessary to put 1 so that the program prints all statistics (at first by lines, and then the general on the text), otherwise only the general text is printed.

Selective analysis of text by lines

If you execute a command

```
propertiesTextG3 <path to the file with Ukrainian text for analysis> @n
```

then the text from the file will be displayed on the screen and will be shown with the numbers of all lines to the left of the lines themselves, separated from the text using the tab character (displayed as a space with a non-constant width, which depends on the system settings). Then you can run the same program (you can do without it, but you can specify other numbers than the program will consider) for analysis the selected rows. To do this, to the commands propertiesTextG3, in addition to the last mentioned, anywhere in the command line to the vertical bar (up to pipeline) add the first and last line numbers, separated by a colon (without any other characters, including spaces). You can specify several such pairs, the information will be displayed in the same order. If some line numbers will occur several times, they will be displayed (if this option is specified) and analyzed several times (probably, due to the laziness of Haskell programming language, they can use (if they are not garbage collected till the time of re-usage) the once calculated and memoized results). If text break control is specified on lines equal to 1, then the program will combine and analyze the lines whose numbers were specified and correspond to the numbers when outputting the command from @n.

All this allows you to focus on the text or only part of it.

Working with rewritePoemG3

Verify that the folder (directory) where cabal installed the executables of the programs is available for search in the PATH environment variable.

The program rewritePoemG3 starting from the version 0.12.0.0 of the package can work in several modes: in the multiple properties mode, in the single property mode, in the comparative mode.

Multiple properties mode (+m …-m)

If among the command line arguments there is specified a group of arguments inside the frame of +m and -m delimiters and inside it there are several properties in their encoding (see: Types of properties) delimiting each of them by the space from the previous and the next ones and if there are not specified ”+c” as one of the command line arguments, then the program will work in the multiple properties mode (if the property inside the frame is just one solely, then the program will run in the single property mode, see below).

The syntax of the program work in such a case is the following:

```
rewritePoemG3 <first argument> <file with Ukrainian text> [<whether to use
  “growing lines”>] +m <types of properties> -m <numeric arguments>
```

Upon successful completion of the program (there should be no messages) in the same folder (directory) as the file with the text that is overwritten, there must be files with the additional ending ’.new.txt’ and the prefices delimited with the dot with the properties encoding. It is there that the converted texts (for example, a poem) are written to according to the input data.

The entered data applies to the entire text, to each line of text in particular (after its preliminary processing by the program).

Single property mode

If you do not specify among the command line arguments a frame with the +m and -m delimiters, and if there is no ”+c” specification either, then the program will work in the single property mode (it is also applicable in tne case of just single property specified in the multiple properties mode).

The syntax of the program work in such a case is the following:

```
rewritePoemG3 <first argument > [<whether to grow lines>] <Ukrainian text file>
  <property type> <numerical arguments>
```

You will see something like the following:

```
rewritePoemG3 10.0_1.2 “sadok.txt” yyy 5 1 2
```

Upon successful completion of the program (there should be no messages) in the same folder (directory) as the file with the text that is overwritten, there must be a file with the additional ending ’.new.txt’ and the prefix delimited with the dot with the property encoding. It is there that the converted text (for example, a poem) is written according to the input data.

The entered data applies to the entire text, to each line of text in particular (after its preliminary processing by the program).

More complex usage

While using the multiple properties mode or the single property mode the numeric arguments have much the same meaning as for the lineVariantsG3 program.

Numeric arguments, if specified, have the following meaning.

The first numeric argument is the number of intervals into which the interval between the minimum and maximum value of the property for this line. If not specified, it is considered equal to 1. A value of 0 does not allow other numeric arguments to further change the result of the work of the program.

All subsequent numeric arguments (if specified, otherwise no permutations occur) are interval numbers that will be swapped with the interval with the maximum values of the property. This allows you to change the structure of the data that is displayed as a result of the program and see the internal (not maximum) items. For example, the numeric arguments 6 1 4 (in this order) will mean that during

program execution the interval between the maximum and minimum value of the property will be divided into 6 equal intervals, with the elements that are in the first and fourth, counting from the minimum (interval number 1) will be moved to the maximum number (and property values) interval, and then the line with the maximum value of the property is written to the output file. Values that were in the maximum interval will be moved to the interval with the lowest number among those that are moved to maximum.

Comparative mode of operation (+c)

rewritePoemG3 can also be run in so-called ‘comparative’ mode, when it offers strings (one after the other) of several (no more than 7) specified files and writes the selected (or blank line, if none is selected) to the last file (except those ones). So from several files by their comparison by lines you can create a new one. It also allows you to run the program with different properties in the multiple properties mode, then run it in the comparative mode on the received files and create in a fairly easy way their combinations - new variants.

Note: If you plan to get more ‘hints’ and recommendations from the program, it is probably easier (and better) to apply an interactive mode of the lineVariantsG3 program with several properties instead, or even the recursive mode of the multiple sources mode of the lineVariantsG3.

To operate in the comparative mode, use the following command:

```
rewritePoemG3 +c <files to read text variants from> <final file>
```

Types of properties

One of the principles of the program is to search among the text options for those for which the maximum is the value of a function called ‘property’ of the text (just simply: property) and is a specific property for lines. The user can choose the property that will be used during program operation (this is done in the command line once during program operation by the geven (or absent respectively) command line argument). The command line argument of the program call can be:

- ‘y0’ - the first property in time, based on ’periods of uniqueness’. The idea is to estimate the number of sounds, or pauses, or phonetic phenomenae (palatalization of consonants), which are between successive appearances of each sound not in one, but in different words, and the total sum of such distances for different words is sought. The greater the value corresponds to the text with a smoother variable phonetic pattern, (probably) depending on the average number of sounds in the ‘period of uniqueness’ it may be easier to speak, otherwise it is (may be) harder to speak; less meaningful - on the contrary - text with more rapid changes in phonetic pattern, possibly with amplification of separate groups of sounds, which is more typical for intonationally selected and / or poetic texts with appealed or highlightened emotions. When using this property (and only it) the first argument of the program call string does not matter (it is ignored by the program).
- ‘0y’ is the first version of the rhythm-only analysis property (semi-empirical), based on the function of rhythm, that uses the durations of sounds that have been synthesized in the mmsyn6ukr-array software package. The rhythm function is inspired by antiquity poetry, where instead of stressed and unstressed syllables rhythmically short and long alternated; also musical destinies for which the main ones are two-part rhythm and three-part rhythm. The function is implemented in such a way as to make it as easy as possible to capture significant emissions subrhythms for two-syllable and three-syllable cases. Using the <first argument> you can change the ratio of these subproperties and, accordingly, to change the property itself.
- ’02y’ is a ’0y’-like property that uses other durations of sounds synthesized by the r-glpk-phonetic-languages-ukrainian-durations.

Probably one of the most accurate in this version of the available for the task of writing rhythmic text. You can create other variations of sound durations using the capabilities of the package r-glpk-phonetic-languages-ukrainian-durations or in some other way.

- ’03y’ is a ’02y’-like property that uses other durations of sounds synthesized by the r-glpk-phonetic-languages-ukrainian-durations package. Probably one of the most accurate in this version of the available for the task of writing the rhythmic text. You can create other variations of sound durations using the capabilities of the package r-glpk-phonetic-languages-ukrainian-durations or in some other way.
- ’04y’ is a ’02y’-like property that uses other durations of sounds synthesized by the r-glpk-phonetic-languages-ukrainian-durations package. These sound durations are derived from data other than 0y, 02y and 03y, so be careful while mixing them in a multiple properties mode.
- ’y’ - a property that calculates the properties of ’y0’ and ’0y’ in a more efficient way than each of them alone, and then multiplies the received data. Gives higher values for lines with a more smoothly changing phonetic pattern and those that are more rhythmic (from the point of view of the property ’0y’). The use of the <first coefficient> internally affects only the sub-property ’0y’.
- ’y2’ is a property similar to ‘y’, but uses the variant with ’02y’ instead of the second sub-property (rhythmicity).
- ’y3’ is a property similar to ‘y’, but uses the variant with ’03y’ instead of the second sub-property (rhythmicity).
- ’y4’ is a property similar to ‘y’, but uses the variant with ’04y’ instead of the second sub-property (rhythmicity).
- ’yy’ - a property that uses the property ’y0’ and ’0y’, and instead of multiplying them, divides the result of the second by the result of the first. Maximized for texts with high rhythmicity (in terms of the ’0y’ property) and grouping of identical sounds into the groups that are closer to each other. The use of the <first coefficient> affects only the sub-property ’0y’.
- ’yy2’ - a property that uses the property ’y0’ and ’02y’, and instead of multiplying them, divides the result of the second by the result of the first. Maximized for texts with high rhythmicity (in terms of the property ’02y’) and grouping of identical sounds in groups closer to each other. The use of the <first coefficient> internally affects only the sub-property ’02y’.
- ’yy3’ - a property that uses the property ’y0’ and ’03y’, and instead of multiplying them, divides the result of the second by the result of the first. Maximized for texts with high rhythmicity (in terms of the property ’03y’) and grouping of identical sounds in groups closer to each other. The use of the <first coefficient> internally affects only the sub-property ’03y’.

- ‘yy4’ - similar to ’yy’ with the difference that instead of ’0y’ is used ’04y’.

  ’Z’ -line

- ’0z’
- ’02z’
- ’03z’
- ’04z’
- ‘z’
- ‘z2’
- ‘z3’
- ‘z4’
- ‘zz’
- ‘zz2’
- ‘zz3’
- ‘zz4’ These properties are similar to the corresponding ones, where z is replaced by y. But they use more complex rhythmic functions derived from the module Languages.Rhythmicity.Factor from the package phonetic-languages-rhythmicity. Carefully use mixed properties in multiple properties mode, because they actually represent different approaches within the general method, so they can give in pairs less compatible results, but when used correctly they give an acceptable result. You may need a little practice, also more often use the propertiesTextG program.

  Besides, while working with the following properties the concept of the polyrhythmicity as a source of rhythmicity is used.

- ”w01” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one maximum value the most significantly influences the rhythmicity and another maximum one influences less. As a variant of the durations calcuation function syllableDurationsD is used;
- ”w02” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one maximum value the most significantly influences the rhythmicity and another maximum one influences less. As a variant of the durations calcuation function syllableDurationsD2 is used;
- ”w03” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one maximum value the most significantly influences the rhythmicity and another maximum one influences less. As a variant of the durations calcuation function syllableDurationsD3 is used;
- ”w04” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one maximum value the most significantly influences the rhythmicity and another maximum one influences less. As a variant of the durations calcuation function syllableDurationsD4 is used;
- ”w11” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the two maximum values the most significantly influence the rhythmicity and another minimum one influences less. As a variant of the durations calcuation function syllableDurationsD is used;
- ”w12” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the two maximum values the most significantly influence the rhythmicity and another minimum one influences less. As a variant of the durations calcuation function syllableDurationsD2 is used;
- ”w13” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the two maximum values the most significantly influence the rhythmicity and another minimum one influences less. As a variant of the durations calcuation function syllableDurationsD3 is used;
- ”w14” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the two maximum values the most significantly influence the rhythmicity and another minimum one influences less. As a variant of the durations calcuation function syllableDurationsD4 is used;

- ”w21” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one minimum value the most significantly influences the rhythmicity and two maximum ones influence less. As a variant of the durations calcuation function syllableDurationsD is used;
- ”w22” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one minimum value the most significantly influences the rhythmicity and two maximum ones influence less. As a variant of the durations calcuation function syllableDurationsD2 is used;
- ”w23” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one minimum value the most significantly influences the rhythmicity and two maximum ones influence less. As a variant of the durations calcuation function syllableDurationsD3 is used;
- ”w24” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one minimum value the most significantly influences the rhythmicity and two maximum ones influence less. As a variant of the durations calcuation function syllableDurationsD4 is used;
- ”w31” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one minimum value the most significantly influences the rhythmicity and another minimum one influences less. As a variant of the durations calcuation function syllableDurationsD is used;
- ”w32” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one minimum value the most significantly influences the rhythmicity and another minimum one influences less. As a variant of the durations calcuation function syllableDurationsD2 is used;
- ”w33” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one minimum value the most significantly influences the rhythmicity and another minimum one influences less. As a variant of the durations calcuation function syllableDurationsD3 is used;
- ”w34” – more complex rhythmic structure with some another way of determining the property value, it takes the string as the text with rhythmic groups of 4 syllables each where the one minimum value the most significantly influences the rhythmicity and another minimum one influences less. As a variant of the durations calcuation function syllableDurationsD4 is used;

- ”x01” – similarly to the “w01”, but with more complex dependency for the less significant duration and probably less prognosable results;
- ”x02” – similarly to the “w02”, but with more complex dependency for the less significant duration and probably less prognosable results;
- ”x03” – similarly to the “w03”, but with more complex dependency for the less significant duration and probably less prognosable results;
- ”x04” – similarly to the “w04”, but with more complex dependency for the less significant duration and probably less prognosable results;

The following values are similar to the corresponding “w” with more complex dependency (as just described ones). Among them:

- ”x11”
- ”x12”
- ”x13”
- ”x14”
- ”x21”
- ”x22”
- ”x23”
- ”x24”
- ”x31”
- ”x32”
- ”x33”

- ”x34”

  If this argument is as follows, then a polyrhythmic analysis of the text is used. More complex properties of the text are searched and checked using more comprehensive by structure properties. This is a research direction in the programs and library usage. Besides there is a possibility also to specify your own custom configuration using the ‘c’, ‘C’, ‘N’ modes. Therefore, the following is used:

- ”u01” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 and even less highlighted 1 maximae, the groups of 5 syllables. For syllable durations is used syllableDurationsD;

- ”u02” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 and even less highlighted 1 maximae, the groups of 5 syllables. For syllable durations is used syllableDurationsD2;

- ”u03” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 and even less highlighted 1 maximae, the groups of 5 syllables. For syllable durations is used syllableDurationsD3;

- ”u04” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 and even less highlighted 1 maximae, the groups of 5 syllables. For syllable durations is used syllableDurationsD4;

- ”u11” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 maximae and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD;

- ”u12” □ A polyrhythm with the most highlighted 1 maximum, less highlighted 2 maximae and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD2;

- ”u13” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 maximae and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD3;

- ”u14” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 maximae and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD4;

- ”u21” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 minimae and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD;

- ”u22” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 minimae and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD2;
- ”u23” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 minimae and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD3;
- ”u24” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 minimae and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD4;
- ”u31” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 minimae and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD;
- ”u32” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 minimae and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD2;
- ”u33” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 minimae and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD3;
- ”u34” -> A polyrhythm with the most highlighted 1 maximum, less highlighted 2 minimae and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD4;
- ”u41” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 maximum and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD;
- ”u42” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 maximum and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD2;
- ”u43” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 maximum and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD3;
- ”u44” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 maximum and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD4;

- ”u51” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 maximum and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD;
- ”u52” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 maximum and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD2;
- ”u53” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 maximum and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD3;
- ”u54” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 maximum and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD4;
- ”u61” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 minimum and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD;
- ”u62” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 minimum and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD2;
- ”u63” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 minimum and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD3;
- ”u64” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 minimum and even less highlighted 1 maximum, the groups of 5 syllables. For syllable durations is used syllableDurationsD4;
- ”u71” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 minimum and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD;
- ”u72” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 minimum and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD2;
- ”u73” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 minimum and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD3;

- ”u74” -> A polyrhythm with the most highlighted 2 maximae, less highlighted 1 minimum and even less highlighted 1 minimum, the groups of 5 syllables. For syllable durations is used syllableDurationsD4;
- ”v01” and other “v” – are analogous to the corresponding “u” lines with that difference that they use only increasing function variants for the rhythmicity estimation. This makes them the more straightforward ones.
- ”s01” and other “s” – are analogous to the corresponding “u” lines with that difference that they group syllables into groups of 6, not 5.
- ”t01” and other “t” lines – are analogous to the corresponding “s” lines with that difference that they only use increasing function variants for the rhythmicity estimation. This makes them the more straightforward ones.
- “S” line corresponds to the properties of the “s” line with the difference that the following function is used: i. e. rhythmicityPoly-WeightedF2;
- “T” line corresponds to the properties of the “t” line with the difference that the following function is used: i. e. rhythmicityPoly-WeightedF20;
- “U” line corresponds to the properties of the “u” line with the difference that the following function is used: i. e. rhythmicityPoly-WeightedF2;
- “V” line corresponds to the properties of the “v” line with the difference that the following function is used: i. e. rhythmicityPoly-WeightedF20;
- “W” line corresponds to the properties of the “u” line with the difference that the following function is used: i. e. rhythmicityPoly-WeightedF3;
- “X” line corresponds to the properties of the “v” line with the difference that the following function is used: i. e. rhythmicityPoly-WeightedF30;
- “Y” line corresponds to the properties of the “s” line with the difference that the following function is used: i. e. rhythmicityPoly-WeightedF3;

- “Z” line corresponds to the properties of the “t” line with the difference that the following function is used: i. e. rhythmicityPolyWeightedF30;

  The following property lines try to increase the significance of the text ending and decrease the significance of its beginning.

- “I” line corresponds to the properties of the “W” line with the difference that the following weighted function is used: rhythmicityPolyWeightedEF2;
- “J” line corresponds to the properties of the “X” line with the difference that the following weighted function is used: rhythmicityPolyWeightedEF20;
- “K” line corresponds to the properties of the “Y” line with the difference that the following weighted function is used: rhythmicityPolyWeightedEF2;
- “L” line corresponds to the properties of the “Z” line with the difference that the following weighted function is used: rhythmicityPolyWeightedEF20;
- “O” line corresponds to the properties of the “U” line with the difference that the following weighted function is used: rhythmicityPolyWeightedEF3;
- “P” line corresponds to the properties of the “V” line with the difference that the following weighted function is used: rhythmicityPolyWeightedEF30;
- “Q” line corresponds to the properties of the “S” line with the difference that the following weighted function is used: rhythmicityPolyWeightedEF3;
- “R” line corresponds to the properties of the “T” line with the difference that the following weighted function is used: rhythmicityPolyWeightedEF30;

  Starting from the version 0.10.0.0 there are introduced also the following properties:

- “o” line corresponds to the properties of the “u” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLF2;

- “p” line corresponds to the properties of the “v” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLF20;
- “q” line corresponds to the properties of the “s” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLF2;
- “r” line corresponds to the properties of the “t” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLF20;
- “k” line corresponds to the properties of the “u” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLF3;
- “l” line corresponds to the properties of the “v” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLF30;
- “m” line corresponds to the properties of the “s” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLF3;
- “n” line corresponds to the properties of the “t” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLF30;

  The following lines try to increase the importance of the line ending and to decrease the importance of the its beginning.

- “g” line corresponds to the properties of the “u” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLEF2;
- “h” line corresponds to the properties of the “v” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLEF20;
- “i” line corresponds to the properties of the “s” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLEF2;
- “j” line corresponds to the properties of the “t” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLEF20;

- “b” line corresponds to the properties of the “u” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLEF3;
- “d” line corresponds to the properties of the “v” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLEF30;
- “e” line corresponds to the properties of the “s” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLEF3;
- “f” line corresponds to the properties of the “t” line with the difference that the following weighted function is used: rhythmicityPolyWeightedLEF30;

Custom configuration properties

If the property type starts with one of the “c”, “A”, “B”, “C”, “D”, “E”, “F”, “M”, “N” then the program tries to parse this property as a encoded configuration of the polyrhythmicity. A mode for the developers and researchers, there can be used even more complex polyrhythmic structures. For example, ”c114+112=2” returns as a polyrhythm structure data P1 (Ch 1 1 4) (Rhythm 1 1 2) 2, that means that the 1 most highlighted maximum, and 1 less highlighted maximum is searched in the groups of 4 syllables using syllableDurationsD2; ”cttttff7+112111=7*3” returns as a polyrhythm structure data P2 (PolyCh [True,True,True,False,False] 7) (PolyRhythm [1,1,2,1,1,1]) 7 3, that means that the 1 most highlighted maximum, and 1 less highlighted maximum, and 2 even less highlighted maximae, and 1 even less highlighted minimum, and 1 even less highlighted minimum in the groups of 7 syllables using the syllableDurationsD3 etc.

While using the new properties of the “A”, “B”, “C”, “D”, “E”, “F”, “M” and “N” lines there will be used ‘weighted’ functions (highly experimental ones, though hopefully effective) that try to take into account also the significance of the place of the line part e. g. internally there will be used such functions as rhythmicityPolyWeightedLEF2, rhythmicityPolyWeightedEF2, rhythmicityPolyWeightedF2, rhythmicityPolyWeightedLF2, rhythmicityPolyWeightedLEF3, rhythmicityPolyWeightedLF3, rhythmicityPolyWeightedEF3 and rhythmicityPolyWeightedF3 respectively, all from the module Rhythmicity.PolyRhythm.

Tip: If there are no results in the program output using multiple properties (an empty output), increase the number of groups in properties (for at least one) and / or add interval transforms to change the internal structure of certain properties.

The first argument

For the first time the program can be used without this argument, or by entering in its place 1_. You may want to in the future to deepen the analysis. Then you can also specify the first argument (it is the first in the list of command line arguments, not counting the group - if available - between arguments +a and -a) in the form number1_number2, where number1 and number2 - decimal positive fractions or positive integer numbers, and one of them may be absent (then it is considered equal to 1). Example,

3.4_2 2.987_0.7865 0.0001_ etc.

The first number is a coefficient multiplied by the component of the property that is responsible for the rhythmicity of the two-syllables based verse system, and the second – for rhythmicity of three-syllables based verse system. Accordingly, a successful combination can emphasize, combine or eliminate, reduce the effect of rhythm for two-component or three-component based verse system.

In the absence of this argument, the programs behave as if it were equal to 1_1.

64

Constraints

When you run the lineVariantsG3 program you can specify as command arguments constraints string. They allow to reduce the number of calculations, to consider only certain options (for example, with a certain definite order of some words, etc.) that allows you to actually expand the program capabilities. These limitations are encoded as command arguments line between two special notation 'brackets' '+a' and '-a'. They form a group of arguments that can stand anywhere in the input line data. Depending on these arguments, the program asks or does not ask an additional question for verification and confirmation (that is called double check).

There are 6 types of constraints, they can be arbitrarily combined, but with respect to boundaries for each of them.

The figure shows that all types implemented with one argument that is the same for all of them – the number of words (or their combinations) in a line. The user, having started the program, can no longer adjust during its work this amount, but it is important for limitations in general. None of the digital characters in the constraints should be greater than this number, also this number is not more than 6 and not less than 0. Also a necessary condition is that no numeric characters within one encoded constraint cannot be repeated twice. For example, the following constraints are obviously not valid: Q2235 (repetition of digits), E2 (digital characters where they do not exist), T247 (7 is greater than 6), F0 (one character instead of the required two), A37523 (7 is greater than 6), B5 (one symbol, and there must be other(s) one(s)). Incorrectly set constraints will either not affect the result (although it will be expected otherwise), or will cause runtime exception and program shutdown. Since the result of their application is not simple, so the program at its work displays a line to which the entered constraints will be applied with an additional question, whether all data are entered correctly.

The types of costraints and their values are given in more detail in the table.

- Constraint E – Without entering additional digital characters – Corresponds to the absence of additional constraints, so does not affects the end result.
- Constraint Q – 4 pairs of unequal digits in range from 0 to the number of words or their comcatenations minus 1 – Numbers are

indices of 4 words or their concatenations, the mutual order of which during the permutations will be saved as follows. Also, if these words are the same (excluding uppercase and lowercase letters), then it is a convenient way to reduce the amount of data to be analyzed.

- Constraint T – 3 pairs of unequal digits in range from 0 to the number of words or their concatenations minus 1 – Numbers are indices of 3 words or their concatenations, the mutual order of which during the permutations will be saved as follows. Also, if these words are the same (excluding uppercase and lowercase letters), then it is a convenient way to reduce the amount of data to be analyzed.
- Constraint F – 2 pairs of unequal digits in range from 0 to the number of words or their combinations minus 1 – Numbers are indices of 2 words or their concatenations, the mutual order of which during the permutations will be saved as follows. Also, if these words are the same (excluding uppercase and lowercase letters), then it is a convenient way to reduce the amount of data to be analyzed.
- Constraint A – 1 digit and a few more in pairs unequal numbers (all among themselves unequal) to the right of it within – The first digit is the index of the element relative to which the placement of all other elements is determined (words or their combinations); all other numbers on the right are indices of the elements that should stand in the resulting permutations to the RIGHT of the element with from the element with the index equal to the first digit.
- Constraint B – 1 digit and a few more in pairs unequal numbers (all among themselves unequal) to the right of it within – The first digit is the index of the element relative to which the placement of all other elements is determined (words or their combinations); all other numbers on the right are indices of the elements that should stand in the resulting permutations to the LEFT of the element with from the element with the index equal to the first digit.

Parallel execution of programs

Typically, all packet programs run on a single processor core. In this case, for all programs under consideration, there is an opportunity to enable multi-core operation - parallel computing. To do this, the command line arguments must include the following:

```
+RTS -N -RTS
```

Their placement does not affect the order and value of other command line arguments, and there may be other RTS entries parameters. For more information on these parameters, see the documentation [18].

We can only recommend these settings for the propertiesTextG3 program. For other programs, they are not recommended, although you can use them (they will simply increase the use of resources).

68

Bibliography

[1] ВД Шарко. Актуальні проблеми природничо-математичної освіти в середній і вищій школі.

[2] Наталія Костенко. ЕЛЕМЕНТИ ТОНІЧНОГО І СИЛАБІЧНОГО ВІРШУВАННЯ В ДУМОВОМУ ВІРШІ ТГ ШЕВЧЕНКА. *НАШ УКРАЇНСЬКИЙ ДІМ*, page 38.

[3] О. В. Лазер-Паньків та ін. Л. Л. Звонська, Н. В. Корольова. Ямбічна строфа // Енциклопедичний словник класичних мов., 2017.

[4] АГ Кошовий and ГІ Кошовий. Одновимірні самоподібні фрактали та їх використання у моделюванні. 2011.

[5] Під ред. Анатолія Волкова. Айрен // Лексикон загального та порівняльного літературознавства., 2001.

[6] Під ред. Анатолія Волкова. Баяті // Лексикон загального та порівняльного літературознавства., 2001.

[7] Під ред. Анатолія Волкова. Буриме // Лексикон загального та порівняльного літературознавства., 2001.

[8] Під ред. Анатолія Волкова. Варіативність // Лексикон загального та порівняльного літературознавства., 2001.

[9] Під ред. Анатолія Волкова. Верлібр // Лексикон загального та порівняльного літературознавства., 2001.

[10] Під ред. Анатолія Волкова. Адекватний переклад // Лексикон загального та порівняльного літературознавства., 2001.

[11] Під ред. Анатолія Волкова. Александрійський вірш // Лексикон загального та порівняльного літературознавства., 2001.

[12] Під ред. Анатолія Волкова. Алкеєва строфа // Лексикон загального та порівняльного літературознавства., 2001.

[13] Під ред. Анатолія Волкова. Античні розміри // Лексикон загального та порівняльного літературознавства., 2001.

[14] Під ред. Анатолія Волкова. Античні строфи // Лексикон загального та порівняльного літературознавства., 2001.

[15] Під ред. Анатолія Волкова. Асклепіадова строфа // Лексикон загального та порівняльного літературознавства., 2001.

[16] Під ред. Анатолія Волкова. Аруз, або Аруд // Лексикон загального та порівняльного літературознавства., 2001.

[17] Під ред. Анатолія Волкова. Алкманова або Архілохова строфа // Лексикон загального та порівняльного літературознавства., 2001.

[18] Glasgow haskell compiler user's guide. 7.5. using smp parallelism. [Електронний ресурс]. Режим доступу: https://downloads.haskell.org/ghc/latest/docs/users_guide.pdf. Перевірено 10 листопада 2020 р.

[19] Елена Юрьевна Муратова. Синергетический подход к проблеме смыслопорождения в системе поэтического текста. 2009.

[20] Смаглій Г. А. Теорія музики : Підруч. для навч. закл. освіти, культури і мистецтв., 2013.

[21] МИКОЛА ВАСИЛЬОВИЧ Гуцуляк. *Українське віршування 30-80-х рр. XVII ст*. PhD thesis, «Теорія літератури». Чернівці: ЧНУ ім. Юрія Федьковича, 2017, 2017.

[22] Оксана Валентинівна Кудряшова. Functional poetics. 2016.

[23] Тарас Шевченко. *Садок вишневий коло хати*. Strelbytskyy Multimedia Publishing, 2018.

[24] ЕЮ Муратова. Специфика синергетического анализа поэтического текста. 2012.

[25] К Паладян. Початки силабо-тонічної версифікації в румунській літературі. *Питання літературознавства*, (81):164–172, 2010.

[26] ОВ Любімова. Відтворення античних розмірів в українській поезії 80-х–90-х років XIX століття. *Науковий вісник Чернівецького університету. Романо-слов'янський дискурс*, (565):190–193, 2011.

[27] П Івончак. Український силабо-тонічний вірш 50-х років XIX століття. *Науковий вісник Чернівецького національного університету. Слов'янська філологія*, (585-586):80–84, 2012.

[28] Huang Wan-Li. The extremity laws of hydro-thermodynamics. *Applied Mathematics and Mechanics*, 4(4):501–510, 1983.

[29] Andrés E Coca, Gerard O Tost, and Zhao Liang. Controlling chaotic melodies. *Proc. Encuentro Nacional de Investigación en Posgrados (ENIP)*, 2009.

[30] EJ Garba. Music programming–rule-based randomization of melodic patterns. 2008.

[31] Kerri Welch. *A fractal topology of time: Implications for consciousness and cosmology*. California Institute of Integral Studies, 2010.

[32] David Avnir, Ofer Biham, Daniel Lidar, and Ofer Malcai. Is the geometry of nature fractal? *Science*, 279(5347):39–40, 1998.

[33] Dietmar Saupe. Algorithms for random fractals. In *The science of fractal images*, pages 71–136. Springer, 1988.

[34] Jun Kigami. *Analysis on fractals*. Number 143. Cambridge University Press, 2001.

[35] Jan Andres, Jiří Fišer, Grzegorz Gabor, and Krzysztof Leśniak. Multivalued fractals. *Chaos, Solitons & Fractals*, 24(3):665–700, 2005.

[36] Jan Andres and Miposlav Rypka. Multivalued fractals and hyperfractals. *International Journal of Bifurcation and Chaos*, 22(01):1250009, 2012.

[37] Kevin Merges. Fractals and art. 2005.

[38] Narayan Partap and Renu Chugh. Fixed point iterative techniques–an application to fractals. *International Journal of Research in Mathematics & Computation*, 4(1):1–7, 2016.

[39] Christopher J Bishop and Yuval Peres. *Fractals in probability and analysis*, volume 162. Cambridge University Press, 2017.

[40] Wolfgang Wildgen. Chaos, fractals and dissipative structures in language. or the end of linguistic structuralism. *Gabriel Altmann und Walter A. Koch (Hg.), Systems. New Paradigms for the Human Sciences, de Gruyter, Berlin*, pages 596–620, 1998.

[41] Michael F Barnsley, John E Hutchinson, and Örjan Stenflo. V-variable fractals: fractals with partial self similarity. *Advances in Mathematics*, 218(6):2051–2088, 2008.

[42] Giuseppe Vitiello. The brain is like an orchestra. better yet, it is like a jazz combo, which doesn't need a conductor. *Chaos*, 11(1):2017, 2017.

[43] Oleksandr Zhabenko. dobutoko-poetry. [Електронний ресурс]. Режим доступу: https://hackage.haskell.org/package/dobutokO-poetry-0.8.1.0. Перевірено 09 листопада 2020 р.

[44] Oleksandr Zhabenko. phonetic-languages-simplified-generalized-examples-array. [Електронний ресурс]. Режим доступу: https://hackage.haskell.org/package/phonetic-languages-simplified-generalized-examples-array. Перевірено 23 жовтня 2021 р.

[45] Oleksandr Zhabenko. phonetic-languages-rhythmicity. [Електронний ресурс]. Режим доступу: https://hackage.haskell.org/package/phonetic-languages-rhythmicity. Перевірено 24 серпня 2020 р.

[46] Alex Goldsmith. Synthesising music: exploiting self-similarity using modular forms.

[47] Rohit Sunkam Ramanujam and Bill Lin. Randomized partially-minimal routing on three-dimensional mesh networks. *IEEE Computer Architecture Letters*, 7(2):37–40, 2008.

[48] Shlomo Alexander and Raymond Orbach. Density of states on fractals:«fractons». *Journal de Physique Lettres*, 43(17):625–631, 1982.

[49] Ihor Nabytovych. ФРАКТАЛИ ТА ФРАКТАЛЬНІ СТРУКТУРИ У ХУДОЖНЬОМУ ТЕКСТІ (на прикладі прози Л. Керрола, КС Льюїса та ХЛ Борхеса). *ВІСНИК ЛЬВІВСЬКОГО УНІВЕРСИТЕТУ. Серія іноземні мови*, (18).

[50] John E Hutchinson. Fractals and self similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981.

[51] Robert S Strichartz. Fractals in the large. *Canadian Journal of Mathematics*, 50(3):638–657, 1998.

[52] Benoit B Mandelbrot and Benoit B Mandelbrot. *The fractal geometry of nature*, volume 1. WH freeman New York, 1982.