Applied Informatics
a SpringerOpen Journal

RESEARCH Open Access

Structure sensitive complexity for symbol-free sequences

Cheng-Yuan Liou[1*], Aleksandr A Simak[1] and Jiun-Wei Liou[2]

* Correspondence:
cyliou@csie.ntu.edu.tw
[1]Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
Full list of author information is available at the end of the article

Abstract

The study proposes our extended method to assess structure complexity for symbol-free sequences, such as literal texts, DNA sequences, rhythm, and musical input. This method is based on L-system and topological entropy for context-free grammar. Inputs are represented as binary trees. Different input features are represented separately within tree structure and actual node contents. Our method infers tree generating grammar and estimates its complexity. This study reviews our previous results on texts and DNA sequences and provides new information regarding them. Also, we show new results measuring complexity of Chinese classical texts and music samples with rhythm and melody components. Our method demonstrates enough sensitivity to extract quasi-regular structured fragments of Chinese texts and to detect irregular styled samples of music inputs. To our knowledge, there is no other method that can detect such quasi-regular patterns.

Background

This work introduces general complexity assessment on structure properties for different types of inputs. Input sequences are represented as binary trees, the concept of L-system (Wikipedia 2005) is borrowed to infer rewriting rules and build corresponding context-free grammars, which are used later to assess the complexity score (Kuich 1970). This complexity score is closely related to the notion of entropy (Shannon 1948). Current work is intended to establish a general vision on such kinds of structural complexity assessment.

One initial work in this field focused on the complexity of musical rhythm (Liou et al. 2010), where binary tree representation almost perfectly fits. Later, our proposed method was applied to the complexity of DNA sequences (Liou et al. 2013a, b). From this arose the question of representation: how can other input types be transformed into a binary tree, while keeping the complexity assessment the same? The third study adapted complexity assessment to general texts encoded as symbol-free sequences (Liou et al. 2013a, b). Symbol-free representation was an important milestone—it allowed to extend method for more generic input data, such as Chinese paragraphs. Finally, the study turns back to music with an attempt to reconsider the initial assessment, redefine it, and make method capable of naturally incorporating both musical melody and rhythm.

Complexity assessment

This section provides a generic version of the earlier proposed method for structural complexity assessment (Liou et al. 2010). Our method in the essence remains the same; however, basic data structure and definitions were modified to equip our approach with new capabilities. Also, previous studies paid attention on the equivalence of bracketed strings and binary tree rewriting systems. This study considered it as being already justified, and bracketed strings do not appear in generalized version of the method any more. Instead, we focused on other issues, updating the notation of our formal grammars and proposing a better view on the classification step. It is worth mentioning that all adjustments follow previous conclusions and important statements, as well.

Binary tree

The procedure of transformation from arbitrary input encoding to the binary tree depends on the nature of the input. Despite this, following remains the same: the resulting binary tree reflects and corresponds to the structure of the input. We will not provide exact specifications here on how to transform different kinds of input into corresponding binary trees, but each following section dedicated to one particular kind does provide such necessary explanation in detail.

Our binary tree defined as follow (Fig. 1):

1. Every sub-tree of a binary tree is a binary tree itself;
2. Every node except the root has a parent node;
3. Every node can have exactly two or none child nodes;
4. Every child node is labeled as left or right;
5. Every node can store some content inside.

Using a branching factor of two gives the tree two useful properties—it is relatively simple to maintain and general enough to get in account inputs properties, which are known to be local in linguistics (Gibson 1998) and music (Simonton 1984).

L-system

Every one of these trees can be considered as the result of consecutive development starting from the root. Each development step corresponds to the next tree level, and nodes at any current level are actually the result of development at a previous level.

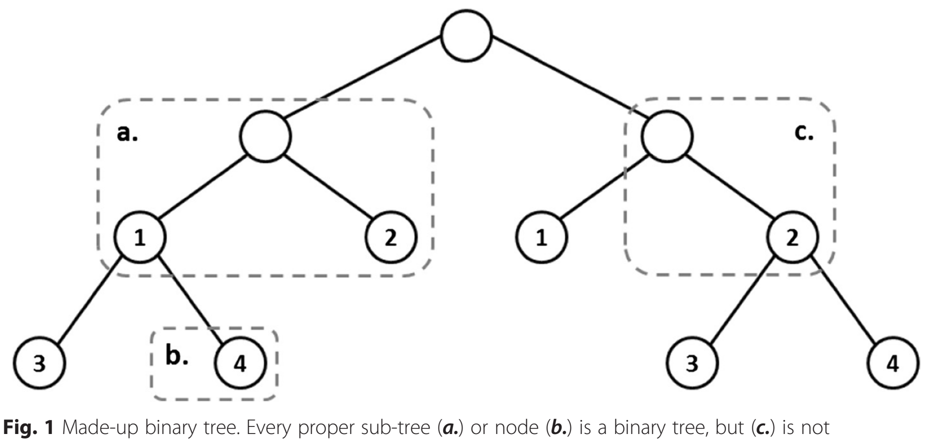


Fig. 1 Made-up binary tree. Every proper sub-tree (***a.***) or node (***b.***) is a binary tree, but (***c.***) is not

The process gradually continues until the original tree is replicated identically. Such development mechanism can be formalized with biology-inspired parallel string rewriting systems, or L-systems (Prusinkiewicz and Lindenmayer 1996). The L-system is a special case of formal grammar (Chomsky 1956). The core of its capabilities is a set of rewriting rules (it explains how every element shall be certainly rewritten), which are applied in parallel, naturalistically reflecting the processes of cell division and plant growth (Lindenmayer 1968). To replicate the tree, it is necessary to construct a complete set of rewriting rules based on labels of the nodes and start the rewriting procedure with the root node as the initial.

Methods

Rewriting rules

Every node in our binary tree, except the root, is labeled denoting whether it is the left or right child. It is necessary to assign a unique label to the root node. Thus, every node in the binary tree shall be labeled. Let symbol *L* states for the left child, *R* states for the right one, and *P* denotes as tree root, all in uppercase as shown in the figure (Fig. 2). Those labels form the set of rewriting system terminal symbols, and their corresponding lowercase symbols *l*, *r*, and *p* are the set of non-terminals. Then, root node non-terminal is the initial starting symbol (or axiom, in formal systems).

Next, for every node in a tree starting with the root, its corresponding rewriting rule is created and placed into a rewriting set one by one (Fig. 3):

1. Left-hand side of rewriting rule contains node non-terminal symbol with the context on a left defined by traversing parent nodes up to the root inclusively and concatenating their labels.
2. Right-hand side of the rule contains node label itself, which is actually a terminal symbol, followed by non-terminals in case the node has a child.
3. An additional operation of node content setting denoted by brackets at right-hand side of the rule immediately after the terminal symbol with the content supposed to be placed inside the node at rewriting moment.

List 1 demonstrates the rewriting rules set for this particular binary tree (Fig. 2) after the procedure above is completed.

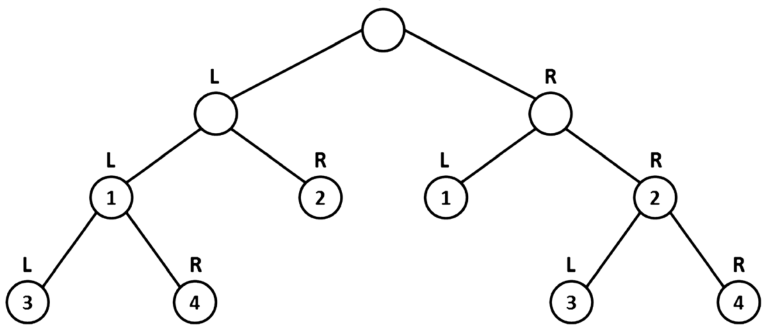


Fig. 2 Properly labeled binary tree

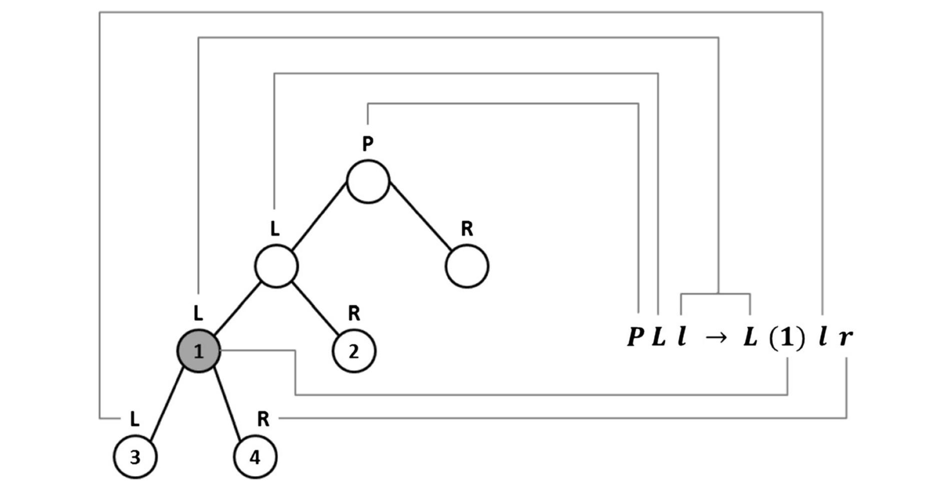


Fig. 3 Rewriting rule creation for dashed node with value 1. List 1. Rewriting rules inside the rewriting set for binary tree from Fig. 2

Thus, such parallel rewriting system is a non-ambiguous context-sensitive formal grammar, which is capable of replicating the original tree identically (Chomsky 1959).

Homomorphism and isomorphism

Curious reader may note two things. Firstly, for every node in a binary tree, there is exactly one corresponding rewriting rule. Secondly, some rewriting rules are quite similar and may appear redundant. The last claim is also correct relative to the tree nodes and even sub-trees. Indeed, some sections of a binary tree may share exactly the same structure and even the same placement of node content. To extract such repeated structures based on their similarity and bound the redundancy of rewriting set, two auxiliary definitions are provided:

Homomorphism in rewriting rules

Two rewriting rules are homomorphic if and only if they assign equal contents to their terminals.

In terms of a binary tree, it means that after the rewriting procedure has been completed, homomorphic nodes share the same content.

Isomorphism on level X in rewriting rules

Two rewriting rules are isomorphic on depth X if and only if they are homomorphic and rules corresponding to their non-terminals are relatively isomorphic on depth X-1. Isomorphism on level 0 indicates homomorphism.

After the rewriting has been completed, two sub-trees of a binary tree are considered isomorphic (on depth X) if their root nodes share the same content and their descendants form an equal structure and relatively share the same content (up to depth X-1).

It is possible to classify all rewriting rules from list 1 using a certain level of isomorphism (Table 1).

It is good to place boundaries on isomorphism depth. Obviously, the lower bound of isomorphism domain is 0 while the upper bound is the number of levels of the

Table 1 Classified rewriting rules with respect to isomorphism levels

Class	Homomorphism	Isomorphism-1	Isomorphism-2
1	$p \mapsto Plr$		
	$Pl \mapsto Llr$	$p \mapsto Plr$	$p \mapsto Plr$
	$Pr \mapsto Rlr$		
2	$PLl \mapsto L(1)lr$	$Pl \mapsto Llr$	$Pl \mapsto Llr$
	$PRl \mapsto L(1)$	$Pr \mapsto Rlr$	
3	$PLr \mapsto R(2)$	$PLl \mapsto L(1)lr$	$Pr \mapsto Rlr$
	$PRr \mapsto R(2)lr$		
4	$PLLl \mapsto L(3)PRRl \mapsto L(3)$	$PLr \mapsto R(2)$	$PLl \mapsto L(1)lr$
5	$PLLr \mapsto R(4)PRRr \mapsto R(4)$	$PRl \mapsto L(1)$	$PLr \mapsto R(2)$
6		$PRr \mapsto R(2)lr$	$PRl \mapsto L(1)$
7		$PLLl \mapsto L(3)$	$PRr \mapsto R(2)lr$
		$PRRl \mapsto L(3)$	
8		$PLLr \mapsto R(4)$	$PLLl \mapsto L(3)$
		$PRRr \mapsto R(4)$	$PRRl \mapsto L(3)$
9			$PLLr \mapsto R(4)$
			$PRRr \mapsto R(4)$

original binary tree. However, such isomorphism depth bounds are quite meaningless. The lower bound does not involve any structural information, while the upper bound does not leave anything to compare with the whole tree. Thus, the meaningful lower and upper for the rewriting rules of isomorphism depth are 1 and depth of the original tree minus 1.

Table 2 Final rewriting set after the classification is finished, rules positions are corresponding to Table 1

Class	Homomorphism	Isomorphism-1	Isomorphism-2
1	$C_1 \mapsto C_1C_1$		
	$C_1 \mapsto C_2C_3$	$C_1 \mapsto C_2C_2$	$C_1 \mapsto C_2C_3$
	$C_1 \mapsto C_2C_3$		
2	$C_2 \mapsto C_4C_5$	$C_2 \mapsto C_3C_4$	$C_2 \mapsto C_4C_5$
	$C_2 \mapsto$ null	$C_2 \mapsto C_5C_6$	
3	$C_3 \mapsto$ null	$C_3 \mapsto C_7C_8$	$C_3 \mapsto C_6C_7$
	$C_3 \mapsto C_4C_5$		
4	$C_4 \mapsto$ null	$C_4 \mapsto$ null	$C_4 \mapsto C_8C_9$
	$C_4 \mapsto$ null		
5	$C_5 \mapsto$ null	$C_5 \mapsto$ null	$C_5 \mapsto$ null
	$C_5 \mapsto$ null		
6		$C_6 \mapsto C_7C_8$	$C_6 \mapsto$ null
7		$C_7 \mapsto$ null	$C_7 \mapsto C_8C_9$
		$C_7 \mapsto$ null	
8		$C_8 \mapsto$ null	$C_8 \mapsto$ null
		$C_8 \mapsto$ null	$C_8 \mapsto$ null
9			$C_9 \mapsto$ null
			$C_9 \mapsto$ null

Classification

The classification of rewriting rules is one of the most important steps for structural complexity assessment. It reveals the hidden redundancy of a binary tree to the explicit form, exploiting the redundancy of the corresponding rewriting set.

All isomorphic rewriting rules are labeled with one denoting class label (Table 1). However, such a simple procedure is quite computationally expensive, despite the chosen domain of rewriting rules or tree nodes. The isomorphism check will be repeatedly performed dozens of times on the same inputs, expanding with factor of two for every level of required isomorphism depth. A good illustration is a straightforward implementation of Fibonacci numbers computation.

A more elegant and less computationally expensive way of doing this is to iteratively assign class labels to all tree nodes depending on the node and its child node labels at previous iteration (Fig. 4). It assumes breadth-first node ordering. The first iteration considers only node content and is equal to the 0-depth isomorphism, or homomorphism. Each of new iteration increases the isomorphism level by 1, thus, the total number of iterations is bounded by the depth of the tree (considering also 0th initial iteration).

New class labels (final nodes values) shall be propagated to the corresponding rewriting rules to compose a new rewriting set, for each rule replacing the left-hand side with its class label and the right-hand side with class labels of its children (Table 2). Some rules in the set will have duplicates. Or, alternatively, every rule occurs exactly once but has an associated counter for how many times it actually appears. This information is required for the following complexity assessment. All labels are considered as non-terminal symbols, additional productions to the dedicated terminal symbol shall be added to the set to conform the formality. The initial symbol is obviously a root node class label.

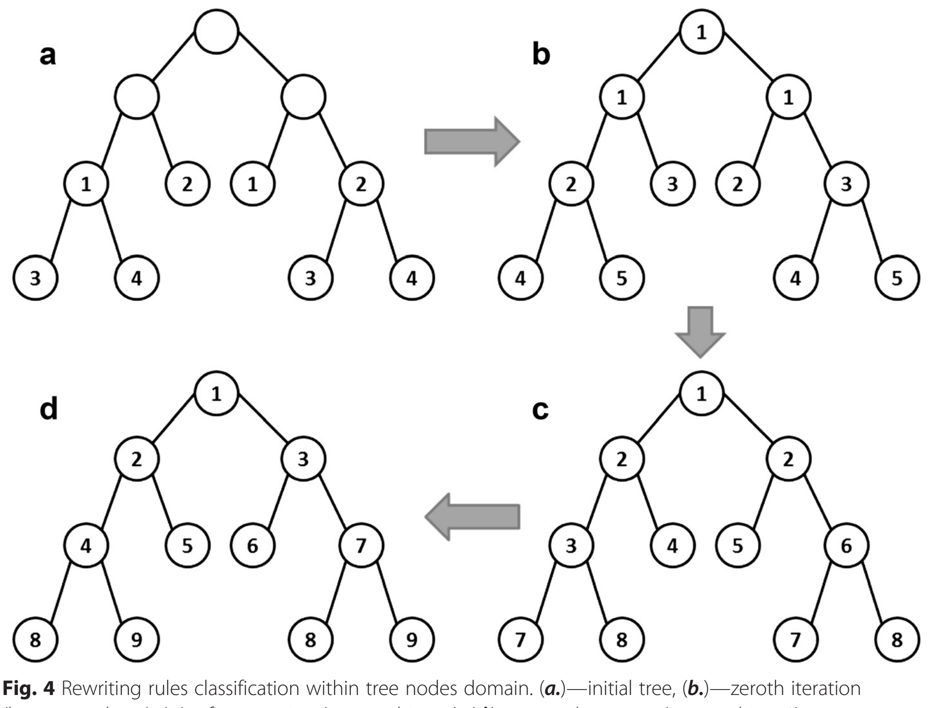


Fig. 4 Rewriting rules classification within tree nodes domain. (***a.***)—initial tree, (***b.***)—zeroth iteration (homomorphism), (***c.***)—first iteration (isomorphism-1), (***d.***)—second iteration (isomorphism-2)

This new parallel rewriting system is a stochastic context-free formal grammar capable of reproducing the original binary tree as well as many other similar trees.

Complexity formula

As mentioned above, a set of classified rewriting rules is a context-free grammar. Thus, the redundancy in the tree (its hidden structure) can be explored by assessing the complexity of tree generating grammar (Liou et al. 2010), which is closely related to the entropy notion for context-free grammars (Kuich, 1970).

The complexity of context-free grammar for binary trees can be evaluated by next three steps:

1. Assume that there are n classes of rules and that each class C_i contains n_i rules. Let $V_i \in \{C_1, C_2, \ldots, C_n\}$, $U_{ij} \in \{R_{ij}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, n_i\}$, and $a_{ijk} \in \{x, x = 1, 2, \ldots, n\}$, where each U_{ik} has the following form:

   $$U_{i1} \rightarrow V_{a_{i11}} V_{a_{i12}}, U_{i2} \rightarrow V_{a_{i21}} V_{a_{i22}}, \ldots \rightarrow \ldots U_{in_i} \rightarrow V_{a_{in_i 1}} V_{a_{in_i 2}}.$$

2. The generating function of V_i, $V_i(z)$ defined as:

   $$V_i(z) = \frac{\sum_{p=1}^{n_i} n_{ip} z V_{a_{ip1}}(z) V_{a_{ip2}}(z)}{\sum_{q=1}^{n_i} n_{iq}},$$

   If V_i does not have non-terminals, set $V_i(z) = 1$.
3. After formulating the generating function $V_i(z)$, we intend to find the largest value of z, $z_{\max}$, at which $V_1(z_{\max})$ still converges (V_1 here denoted the root node rule of a binary tree). After obtaining $z_{\max}$ *of* $V_1(z)$, we set $R = z_{\max}$ (the radius of convergence). We define the complexity of a binary tree as:

   $$K_0 = -\ln R.$$

Numerical estimation

The algorithm for numerical estimation is suggested due to the fact that there is no analytical solution for such a system of complex argument equations. We rewrite generating function and use region tests to approximate the complexity, as follows:

1. Rewrite generating function:

   $$\begin{cases} V_i^m(z') = \dfrac{\sum_{p=1}^{n_i} n_{ip} z' V_{a_{ip1}}^{m-1}(z') V_{a_{ip2}}^{m-1}(z')}{\sum_{q=1}^{n_i} n_{iq}} \\ V_i^0(z') = 1 \end{cases}$$

   and

   $$V_i^0(z') = 1.$$

2. Each iteration, calculate values from $V_i^0(z')$ to $V_i^m(z')$. When $V_i^{m-1}(z') = V_i^m(z')$ for all i, we say V_i^m reaches the convergence for z'. We set $m = 200$.
3. We look up for z'_{max} using dichotomy search to check z' between 0 and 1 for V_i^m convergence.

DNA sequences

In modern bioinformatics, finding an efficient way to locate sequence fragments with biological meaning is an important issue. There are two broadly used categories of methods—sequence complexity (Koslicki 2011) and structure patterns analysis (Manna and Liou 2006; Tino 1998; Peng et al. 1992). Koslicki (2011) presented a method for computing the complexity of a sequence using redefined topological entropy, so the complexity score will not converge to zero for longer sequences. According to Hao et al. (Hao et al. 2000), we can find some rare subsequences by proposed graphical representation for DNA sequences. Zhang and Zhang (1994) analyzed nucleotides occurrence probabilities using four-nucleotide-related functions to draw 3D curves plots.

Our past study gave an attempt on combining statistical and structural properties for input DNA sequences (Liou et al. 2013a, b) within single assessment. We replaced the sequence of four nucleotides with a binary tree and assessed initial sequence complexity, fragmenting the tree to smaller sub-trees and computing the complexity score for each sub-tree independently. The study focused on encoding issue: how to represent a four-nucleotide DNA sequence as a binary tree. We used four fixed tree representations, one for each nucleotide base A, T, C, and G (Fig. 5).

Thus, every input sequence element can be replaced with corresponding tree, and two neighboring trees are combined together under one made-up common root, recursively (Fig. 6).

All of the following steps, such as rewriting rules extraction, classification, and numerical estimation of complexity scores remain the same as stated in the section above.

The study also paid attention to comparing topological entropy (Koslicki 2011) and presented a method of structural complexity, revealing the advanced nature of the latter one. Both methods showed the ability to detect statistical properties of test sequences, but only structural complexity assessment was sensitive to the changes of the sequence sub-words order. In addition, for some input, Koslicki's method cannot compute amino-acid sequences efficiently (required fragment size growths exponentially with sub-word length on alphabet size), but structural complexity does not pose such limitations and can be applied to any amino-acids directly.

The study was successful in attempting to represent symbol sequences as binary trees and encoding sequence symbols with fixed tree structures for the next structural complexity assessment. However, a possible dependency of final complexity scores on chosen fixed representations still was a matter of future study at that moment.

Below we have provided a plot (Fig. 7) of the front part of the structural complexity score for the Zaire Ebola virus (ZEBOV), there are approximately 4000 values, one value for each nucleotide. The whole length of the genome is about 19,000 nucleotides, and it encodes seven structural proteins in the following order: nucleoprotein NP,

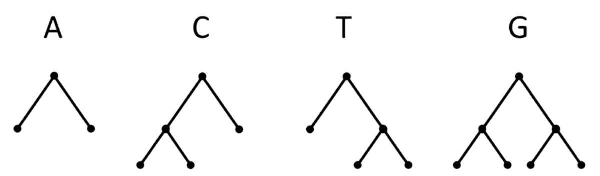


Fig. 5 Fixed tree structures for encoding corresponding nucleotides

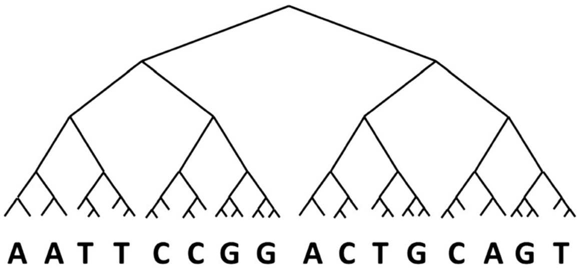


Fig. 6 Complete binary tree for encoded nucleotide sequence

polymerase cofactor VP35, VP40, GP, transcriptional activator VP30, VP24, and RNA polymerase L. The blue and red lines represent two different fragmentation sizes, 64 and 32 nucleotides, respectively. The green dashed lines are what we found at genomic database (U.S. National Library of Medicine), relative to the positions of complexity scores amplitude changes. The first green line and the second green line are the start (470) and the end (2689) positions of nucleoprotein coding sequence (CDS); this segment tends to display a higher complexity with positive gaps and quite short but deep negative spikes. The third green line is a polyA signal (3015…3026) for nucleoprotein, intergenic region (3027…3031), and the transcription start signal (3032…3043) for next polymerase protein complex VP35. The last green line is the beginning of VP35 coding sequence (3129…4151)—the complexity scores return back typically higher values.

Text sequences

Despite successful attempts at encoding input elements with fixed tree structures, two questions were still waiting to be answered:

1. How can we efficiently encode a sequence for alphabet cardinalities higher than the number of nucleotide bases? Encoding every alphabet symbol as fixed tree structure requires deeper trees for larger alphabet symbol sets, and the complexity assessment obviously tends to measure the dependencies between those fixed structures;
2. How do different encodings affect the complexity scores?

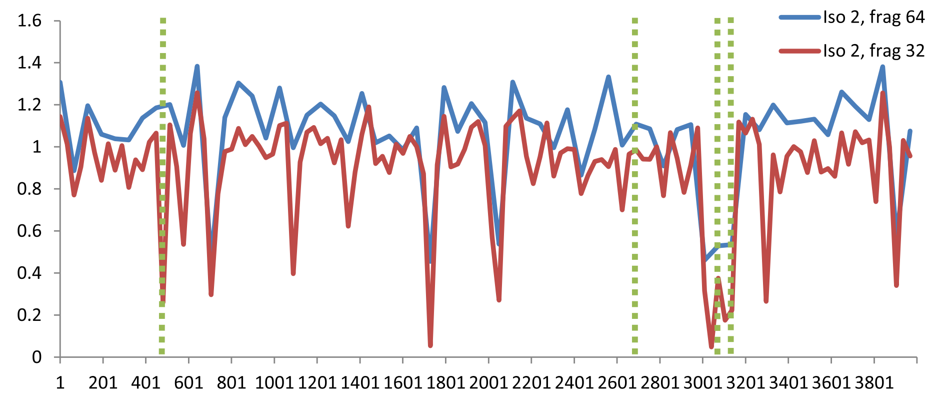


Fig. 7 Zaire Ebola virus complexity scores for 4000 nucleotides, two size segments, isomorphism level 2

Our third study on structure complexity (Liou et al. 2013a, b) addressed both questions. The study concerned regular text sequences, representing natural text as a symbol-free sequence. Symbol-free sequences assume intermediate encoding from symbolic text to binary strings. Two intermediate encodings were used: naive binary (BIN) and advanced Lempel-Ziv-Welch (LZW) (Welch 1984). For English text, BIN encodes 27 alphabet characters (26 Latin letters and space character) directly as a binary representation of the symbol integer index. LZW is a lossless data compression with dictionary-based encoder. LZW saves certain sub-strings from shortest to longest in the dictionary and replaces their occurrences into input sequences with corresponding dictionary indexes. After intermediately encoding every symbol-free binary string, fragments of length 2 were represented with already known fixed tree structures (Fig. 8). Following complexity assessment remains the same.

This study compared sequence complexity for both of the intermediate encodings. Interestingly, the complexity for BIN remains quite uniform over the encoded sequence, while LZW tends to have lower complexity scores in the front and higher scores in the rear of the sequence. Since LZW saves regular patterns in the front part to absorb them later in the rear end, there are not so many regular patterns in the end of the sequence. Also, structural complexity was compared with linguistic complexity (LC) and topological entropy (TE). They also showed similar behavior on intermediate encodings.

The study analyzed intermediate encodings, but some parts of question 2 still remain. Theoretically, there should be no difference in complexity score if all fixed tree replacements are unique, and the replacement procedure is one-to-one function.

However, when we satisfied above two conditions using intermediate BIN encoding and ran the test—our results were surprising (Fig. 9). We tried four different encodings for the same binary string fragments “00,” “01,””10,” and ”11”—corresponding encoding by fixed tree structures denoted by its nucleotide letter.

Later investigation showed that the intermediate encoding BIN encodes 27 symbols as binary strings with length of 5 and fixed tree replacements are aligned to length 2. Original symbols of input sequence became shredded because of this misalignment. Thus, some fixed representation substitutions were formed by ending bit of one symbol and starting bit of the next one. It is not important when one just measures the relative complexity of incoming transmission stream. But when one has to reveal structure complexity of input sequence—such alignment does matter. Since fixed tree representation replaces 2-bit fragments of encoded string—intermediate encoding should be aligned to a multiple of 2.

Chinese texts

In this section, Chinese texts are considered as an extreme case of possible application for structural complexity. Alphabet size or symbol size of such input sequences is of the order

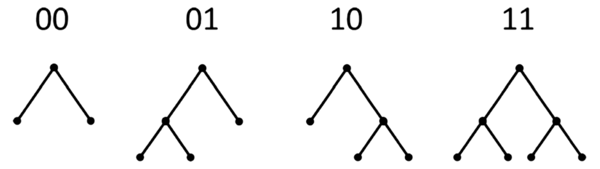


Fig. 8 Fixed tree structures to encode 2-bit segments of binary string

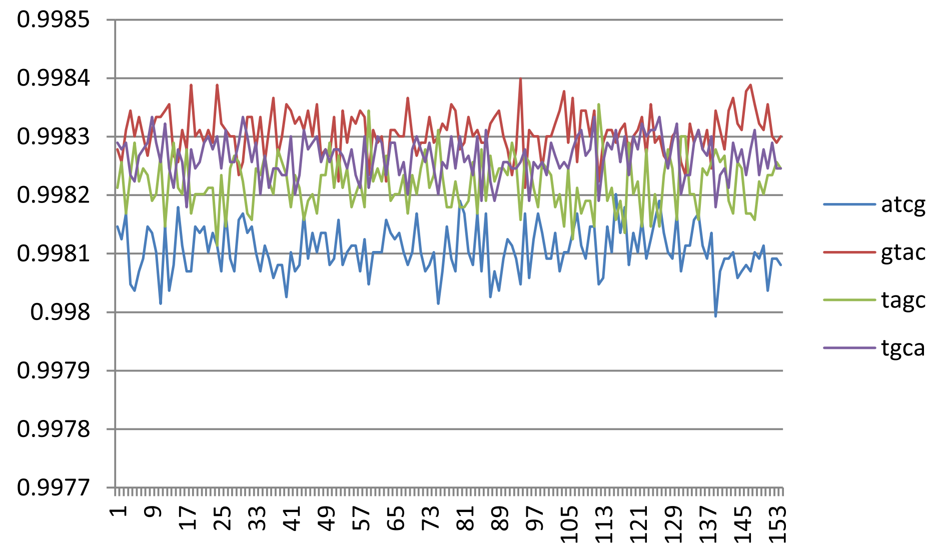


Fig. 9 Four different fixed structures encodings surprisingly reveals different complexity scores

of thousand and can easily exceed the input sequence length. Such alphabet cardinality may also create some restrictions on encoding due to the limitation of memory capacity of modern computers.

Dataset

There are four great classical novels of the Chinese literature (Shep 2011), which are commonly regarded as the greatest and most influential of premodern Chinese fiction. Two of those classical Chinese novels—"Dream of the Red Chamber" (Trad. Chinese "紅樓夢") by Cao Xueqin (18th century) and "Romance of the Three Kingdoms" (Trad. Chinese "三國演義") by Luo Guanzhong (14th century)—were decided for analysis with developed structural complexity method.

Processing

Intermediate encoding of input Unicode symbols (e.g., u4e00, u4e8c) removes the "u" character and considers every 4 hex numbers of two bytes as ASCII symbols, 8 bits each. Thus, all initial input symbols were encoded as 32-bit binary string and concatenated together later. Next, four fixed tree representations were applied to compose binary trees for every of 1024-bit segments of binary input string. Those trees were used as input to perform structural complexity assessments with isomorphism level 8.

Results and discussion

The most fascinating result we have discovered so far is a significantly lower complexity scores for sentences containing regular structures inside. When sentences display a more regular structure than a regular narrative plot (for instance, some poetic inserts), the structural complexity score tends to be lower. Below we provide a few instances of this effect for both novels in descendent order from the highest (less regularity) to the lowest (more regularity) complexity scores. For those who do not feel confident in Chinese, we would recommend paying attention to some regularities in the sequences of the symbols. Some of those regularities are typical for classical Chinese, and some of them are something more.

Dream of the Red Chamber:

1. Chapter 91:
   和他好, 他偏不和你好, 你怎麼樣?你不和他好, 他偏要和你好, 你怎麼
2. Chapter 5:
   「癡情司」,「結怨司」,「朝啼司」,「暮哭司」,「春感司」,「
3. Chapter 1:
   便是『了』,『了』便是『好』;若不『了』便不『好』;若要『好』,
4. Chapter 13:
   、賈敕、賈效、賈敦、賈赦、賈政、賈琮、賈㻞、賈珩、賈珖、賈琛、賈
5. Chapter 54:
   、太婆婆、媳婦、孫子媳婦、重孫子媳婦、親孫子媳婦、姪孫子、重孫子

Romance of the Three Kingdoms:

1. Chapter 20:
   建。建生廣陵侯劉哀。哀生膠水侯劉憲。憲生祖邑侯劉舒。舒生祁陽侯劉
2. Chapter 23:
   也;不讀詩書, 是口濁也;不納忠言, 是耳濁也;不通古今, 是身濁也;
3. Chapter 22:
   之人, 然後有非常之事;有非常之事, 然後立非常之功。夫非常者, 固
4. Chapter 102:
   , 方者為牛腹。垂者為牛舌, 曲者為牛肋。刻者為牛齒, 立者為牛角。細
1. 5. Chapter 20:
   劉昂。昂生漳侯劉祿。祿生沂水侯劉戀。戀生欽陽侯劉英。英生安國侯劉

To our knowledge, there is no other method which can detect such quasi-regular sections.

Music samples

An earlier study (Liou et al. 2010) proposed the complexity measure for musical rhythm, representing it as a binary tree. Such representation seems very natural for rhythm, because notes durations are generally square. The study focused only on the rhythm ignoring another important music component—the melody. Melody gives information on tones transitions through time, specified by rhythm.

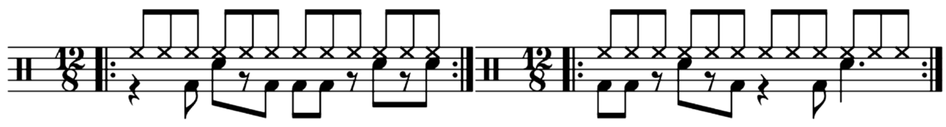
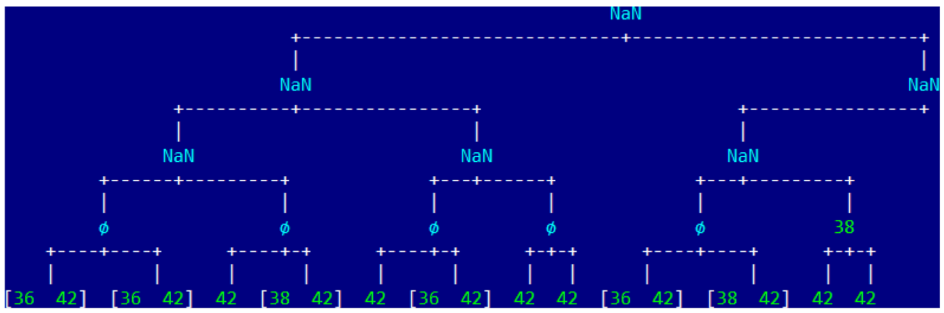


Fig. 10 Blues lesson 57, exercises 6 (*left*) and 7 (*right*)



```
                                             NaN
                 +---------------------------+---------------------------+
                 |                                                       |
                NaN                                                     NaN
          +----------+----------------+                       +----------------+
          |                           |                       |
         NaN                         NaN                     NaN
     +------+---------+          +---+------+           +---+---------+
     |                |          |          |           |             |
     ø                ø          ø          ø           ø             38
  +----+----+      +----+-+   +----+-+     +-+-+     +----+----+     +-+-+
  |         |      |      |   |      |     |   |     |         |     |   |
[36  42]  [36  42]  42  [38  42]  42  [36  42]  42  42  [36  42]  [38  42]  42  42
```

Fig. 11 Multilayered music binary tree for Blues lesson 57 exercises 7. Nodes contain MIDI codes and NAN values are dedicated to keep tree made-up upper part separate

Encoding

This section explains how to incorporate the melody component of music into the assessment. It is the first section where input data have multilayer structure (Fig. 10) and corresponding binary tree representations are truly layered (Fig. 11). This occurs because two beats of rhythmic line can sound at the same time. Hypothetically, for text sequences, it would be so when two characters take one position simultaneously, one character takes more than one position or even both! Binary trees are capable of representing such input by definition. However, there is still an issue of how to bind tonal information into the tree. Representing tonal information with already known fixed tree structures could be a possible solution, but this would cause unexpected difficulty; representing both rhythm and melody with only structural properties of a tree makes them indistinguishable. Later, it causes issues similar to misalignment of data intermediate encodings. The solution we proposed is to keep rhythm within the structure of the binary tree and melody within the content of tree nodes. This section is also novel with the idea to represent different kinds of input features with separate tree properties.

Dataset

We decide to approbate the structural complexity method on a test dataset, the collection of drum lessons for three styles: Rock, Blues, and Jazz. The collection was created and published online by drummer of over 25 years, Rudy Lievens at his personal website (Lievens, 2013) devoted to drums. Exercising materials are provided as note sheets and MP3 or MIDI files for listening and downloading. Exercises download had some issues for few particular files, they were later eliminated from the assessments. In total, after download, Rock had 7533 exercises, and Blues and Jazz had 8594 and 12609 exercises, respectively. Typical lesson note sheets are provided below (Fig. 10).

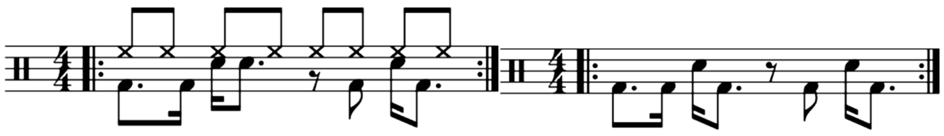


Fig. 12 Rock lesson 205, exercise 1 and exercise 5 (with and without typical hi-hat pattern)

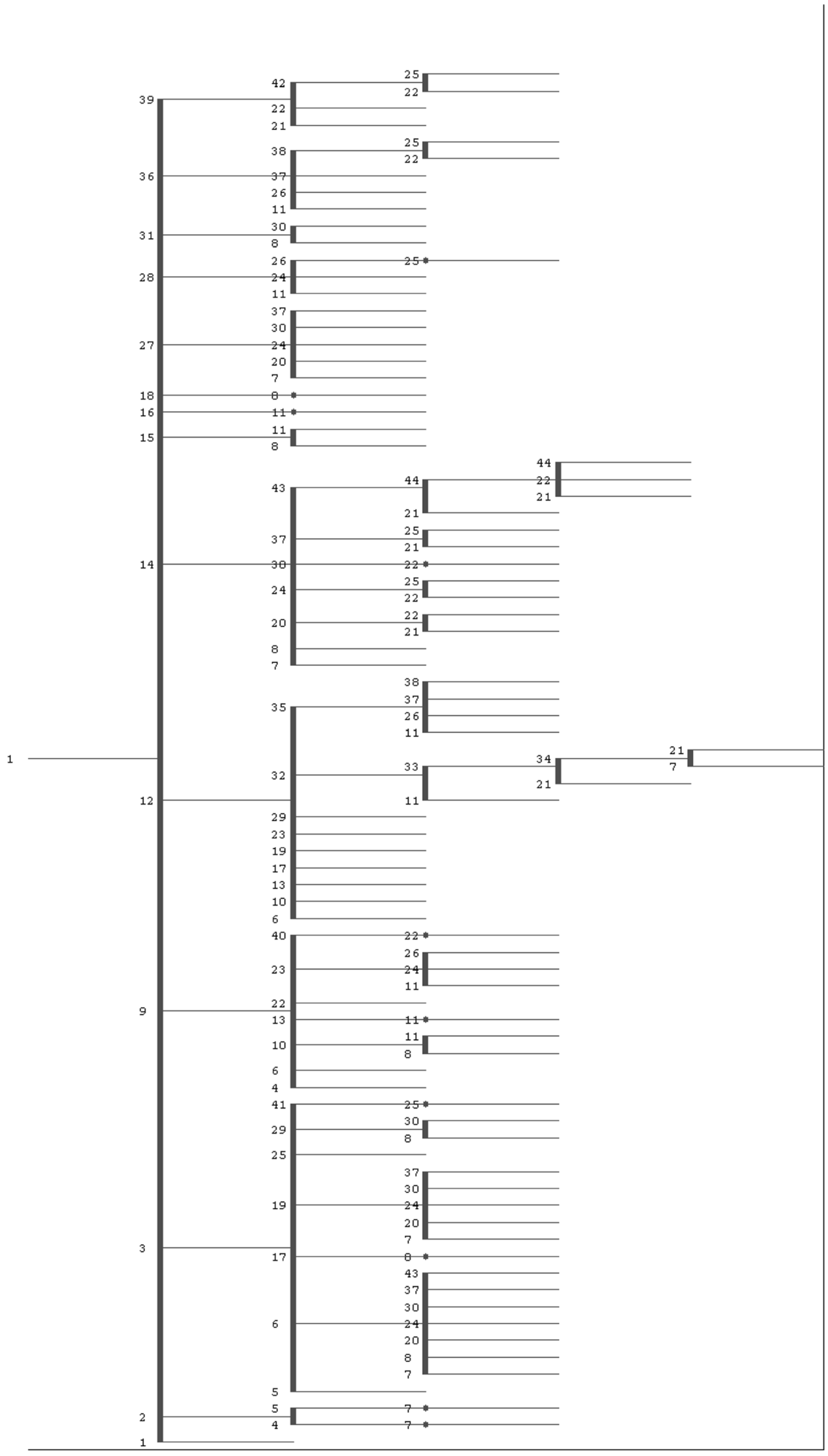


Fig. 13 Rock universal set rules interconnections tree representation, complexity score 2.96

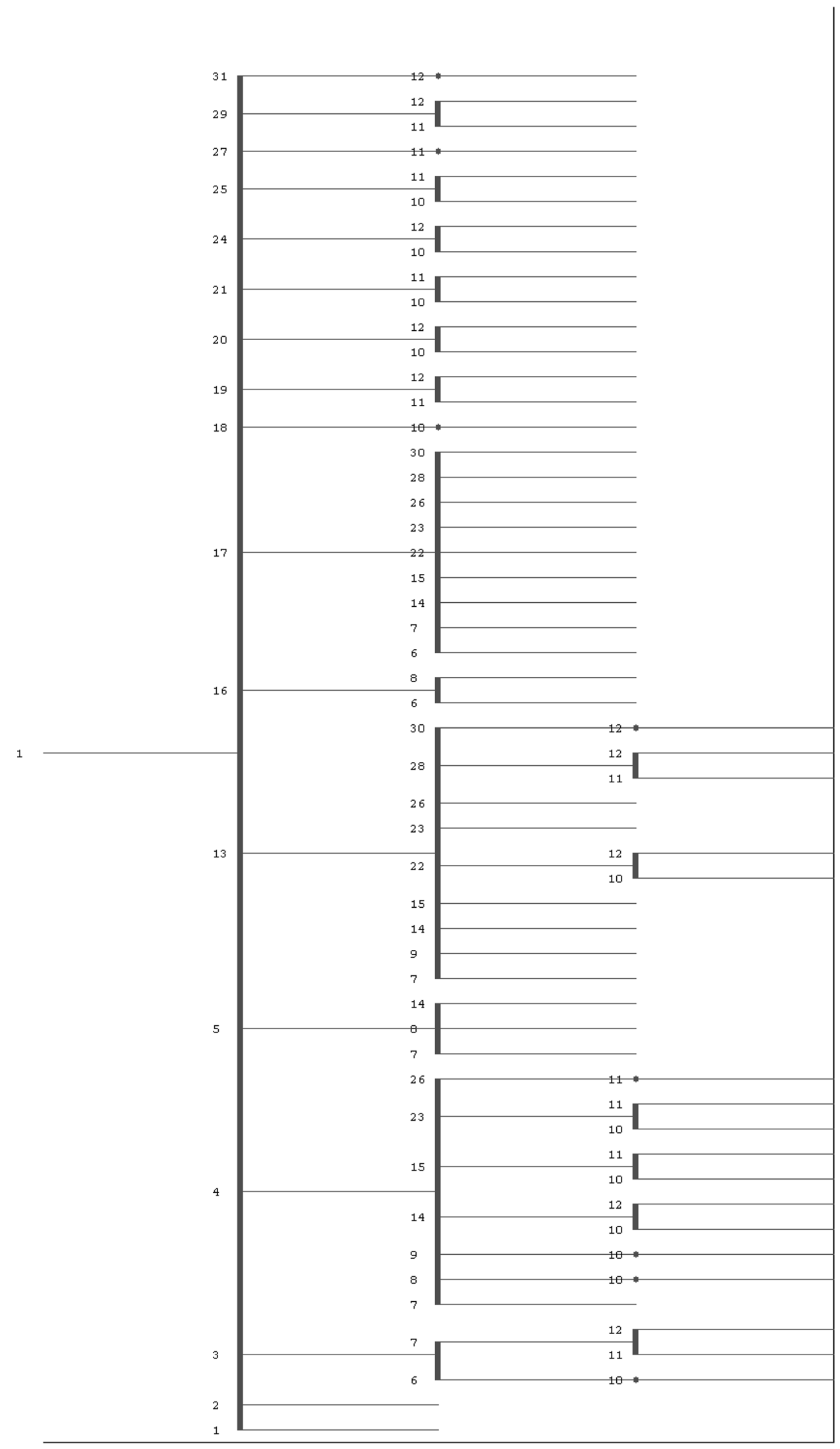


Fig. 14 Blues universal set rules interconnections tree representation, complexity score 2.54

Processing

We conducted preprocessing for all data. The procedure works as uniformly as possible; a single implementation version was used to preprocess all dataset samples. The procedure recognized and properly fixed the following cases: uncertain note onsets and time lags, upbeats and syncopations, and triplets and grace notes. All notes were adjusted to the most suitable positions. Samples with triplets had an additional transformation with multiplication to 3/2 of their durations. Detected grace notes served as indicators to extend their joined notes up to the proper length.

All samples are rather short and structurally similar to each other within one style. Thus, straightforward structural complexity assessment on each sample with isomorphism level 1 does not reveal fascinating results. We decided to assess complexity of each style first and later try to distinguish the most atypical samples within each style. To do so, an additional structure called the universal rewriting rules set is required. This universal rules set contains all rewriting rules from all the samples within one style corpus.

The complexity assessment procedure has been adapted for the current task and was performed in three steps. Step 1 converted preprocessed MIDI files into its binary tree, extracted rewriting rules, and classified them with isomorphism level 1. Step 2 placed classified rewriting rules into universal rules set and accurately maintain their relative probabilities (occurrence scores). The final step assessed the complexity for each sample and each universal set. Numerical estimation of structural complexity for individual samples remains the same, with just one difference—instead of individual rules scores, corresponding scores from universal rules sets were substituted. And to assess the complexity of each style, numerical estimation was applied for each universal rewriting set directly.

Conclusion

The assessment of structural complexity on Rock, Blues, and Jazz universal sets reveals the following scores as 2.96, 2.54, and 3.98, respectively. Also, every set has different numbers of rewriting rules—142, 172, and 688. One might note significant dependency between complexity scores and rewriting sets sizes. For example, Jazz has 688 rewriting rules and the complexity score is dramatically higher. However, higher number of participated in the set rules is not the only necessary component for a higher complexity score. Rules relative probabilities and connections between the rules are actually more important. For example, Blues has 172 rewriting rules, but the complexity score is still significantly lower than Rock with 142 rules. Rock universal set has fewer rules, but they are organized in a more comprehensive way. We tried to illustrate this with two figures (Figs. 13 and 14).

Higher complexity score as well as larger size of universal set for particular corpus might be the direct evidence on a more comprehensive music style. A larger universal set with no dependency on the corresponding complexity score might recall to the richness of music and overall musical expression.

Also, we identified some samples with extremely high complexity. Later examination revealed that they are different from all the other samples of the style. The most evident and easy to understand are several Rock exercises with detected absence of standard for the style hi-hat beats rhythmic line. Figure 12 shows two samples with and without such hi-hat pattern.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
CY conceived the original concept of the study, participated in the design of methods, coordinated the research, and process of drafting the manuscript. AA updated the methods applied in this research, collected and prepared the data for the music analysis, carried out thoroughly the analysis of the music complexity, and drafted the manuscript. JW participated in the analyses of text sequences and has been involved in revising it critically for important content. All authors read and approved the final manuscript.

Acknowledgements
This work was supported by the Ministry of Science and Technology MOST 103-2221-E-002-180 and MOST 104-2811-H-001-004. We also greatly appreciate Rudy Lievens permission to use his data in our research.

Author details
[1]Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan. [2]Institute of Statistical Science, Academia Sinica, Taipei, Taiwan.

Received: 15 April 2015 Accepted: 7 June 2015
Published online: 03 July 2015

References
Chomsky N (1956) Three models for the description of language. IRE Trans Inform Theory 2(3):113–124
Chomsky N (1959) On certain formal properties of grammars. Inf Control 2(2):137–167
Gibson E (1998) Linguistic complexity: locality of syntactic dependencies. Cognition 68(1):1–76
Hao B-L, Lee HC, Zhang S-Y (2000) Fractals related to long DNA sequences and complete genomes. Chaos, Solitons Fractals 11(6):825–836
Koslicki D (2011) Topological entropy of DNA sequences. Bioinformatics 27:1061–1067
Kuich W (1970) On the entropy of context-free languages. Inf Control 16(2):173–200
Lievens, R. (2013). Retrieved 2013, from drum beats, drum lessons and Midi loops: http://www.edrumbeats.com/
Lindenmayer A (1968) Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. J Theor Biol 18(3):280–299
Liou C-Y, Wu T-H, Chia-Ying L (2010) Modelling complexity in musical rhythm. Complexity 15:19–30
Liou C-Y, Liou D-R, Simak AA, Huang B-S (2013a) Syntactic sensitive complexity for symbol-free sequence. In: *LNCS, 4th International Conference, IScIDE 2013, Beijing, China, July 31 – August 2, 8261*. pp 14–21
Liou C-Y, Tseng S-H, Cheng W-C, Tsai H-Y (2013b) Structural complexity of DNA sequence. Comput Math Methods Med 2013:11
Manna S, Liou C-Y (2006) Reverse engineering approach in molecular evolution: simulation and case study with enzyme proteins, Proceedings of the 2006 International Conference on Bioinformatics & Computational Biology, BIOCOMP'06. Las Vegas, Nevada, pp 529–533
Peng C-K, Buldyrev SV, Goldberger A, Havlin S, Sciortino F, Simons M et al (1992) Long-range correlations in nucleotide sequences. Nature 356(6365):168–170
Prusinkiewicz P, Lindenmayer A (1996) The algorithmic beauty of plants. Springer-Verlag, New York
Shannon, C. (1948). The mathematical theory of communication. The Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656, July, October
Shep, S. J. (2011). Paper and print technology. In The Encyclopedia of the Novel, Volume 2 of Wiley-Blackwell Encyclopedia of Literature (p. 596). John Wiley & Sons New Jersey, USA.
Simonton DK (1984) Melodic structure and note transition probabilities: a content analysis of 15,618 classical themes. Psychol Music 12:3–16
Tino P (1998) Spatial representation of symbolic sequences through iterative function systems. Systems Man Cybernetics A 29(4):386–393
U.S. National Library of Medicine. (n.d.). Retrieved from National Center for Biotechnology Information: http://www.ncbi.nlm.nih.gov/
Welch TA (1984) A technique for high-performance data compression. Computer 17(6):8–19
Wikipedia. (2005). *L-system*. Retrieved October 1, 2013, from Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/L-system
Zhang R, Zhang C (1994) Z curves, an intuitive tool for visualizing and analyzing the DNA sequences. J Biomol Struct Dyn 11(4):767–782