FAST NCRNA IDENTIFICATION TECHNIQUES

By

Seyedeh Shohreh Takyar

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Computer Science

2012

ABSTRACT

FAST NCRNA IDENTIFICATION TECHNIQUES

By

Seyedeh Shohreh Takyar

Many noncoding RNAs (ncRNAs) function through both their sequences and secondary structures. Thus, secondary structure derivation is an important issue in today's RNA research. The state-of-the-art structure annotation tools are based on comparative analysis, which derives consensus structure of homologous ncRNAs. Despite promising results from existing ncRNA aligning and consensus structure derivation tools, there is a need for more efficient and accurate ncRNA secondary structure modeling and alignment methods.

In this thesis, we introduce grammar string, a novel ncRNA secondary structure representation that encodes an ncRNAs sequence and secondary structure in the parameter space of a context-free grammar (CFG). Being a string defined on a special alphabet constructed from a CFG, it converts ncRNA alignment into sequence alignment with n square complexity. We explain how this representation is used in derivation of consensus secondary structure through multiple ncRNA alignment and also how existing clustering methods could be applied to ncRNAs represented by this model.

This thesis is dedicated to my parents, Touba and Shahrokh who taught me the most important lesson in my life: Honesty.

ACKNOWLEDGEMENTS

It would not have been possible to write this thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

First and foremost, I would like to express my great appreciation to my advisor, Professor Yanni Sun, for her valuable guidance and advice over the course of this thesis. She has always given me enough time to discuss ideas and evaluate different alternatives in all steps of my research work.

My most sincere appreciation goes to Professor Titus Brown, as both my teacher and as a member of my dissertation committee. I really enjoyed the Open Problem in Bioinformatic course taught by him. He taught me that without having strong biology background, a computer scientist could never make a great stride in Bioinformatics.

I would also like to extend my deepest gratitude to Professor James Cole, from Microbial Ecology Center, who kindly accepted to serve on my examining committee. I really appreciate great comments and suggestions by him.

My special thanks must go to Professor Abdol-Hossein Esfahanian, Associate Chair at the Department of Computer Science and Engineering. He has always supported me, whenever I needed his help to meet different milestone events.

I wish to express my heartfelt thanks to great friends at Michigan State University, for being the surrogate family during the years I lived at Michigan. I will definitely miss our study hours and other gatherings at MSU Dairy Store.

Finally, I am forever indebted to my parents, Touba and Shahrokh, and my family for their endless supports and sacrifices.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

DNA (Deoxyribonucleic acid), RNA (Ribonucleic acid), and proteins are three major essential molecules for all known forms of life. DNA is a double-stranded nucleic acid which carries the entirety of genetic information (called genome) of a living organism. Genome includes thousands of genes and non-coding sequence segments. During a biological process called transcription, genome is copied from DNA into a single stranded nucleic acid molecule called RNA. Transcribed DNA genes are transformed into protein molecules during another biological process called translation and serve as enzymes, hormones, or antibodies for the proper functioning of an organism. However, transcribed non-coding sequences do not have the potential to be translated into protein molecules. These function directly as RNA molecules and are named non-coding RNAs (ncRNA). Figure 1.1 shows the simplified model of **Central Dogma** [5].

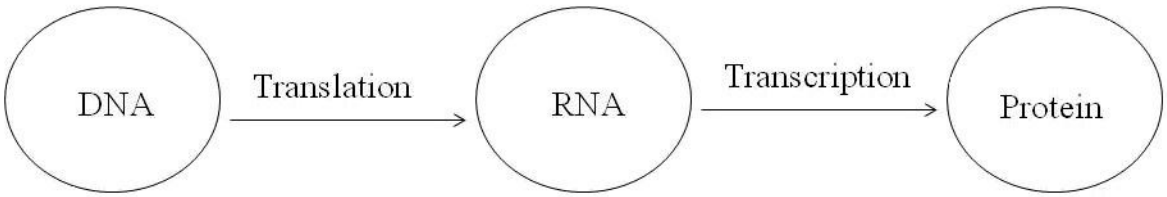


Figure 1.1: Central Dogma schema

1.1 Non-coding RNAs, Structure and functionalities

NcRNA is a single-stranded chain of nucleotides Adenine (A), Uracil (U), Cytosine (C), and Guanine (G). There may also be a hydrogen band between some nucleotide pairs of {A, U}, {C, G}, and {G ,U}, causing nucleotides of each pair to be bounded together and making local folds in ncRNA sequences. These paired-up nucleotides are called **Complementary base pairs**. The two-dimensional shape of ncRNA formed by these complimentary base pairs is called secondary structure of that ncRNA molecule. Figure 1.2 shows the general form of secondary structure and the corresponding names of possible sub-structures made by complimentary base pairs.

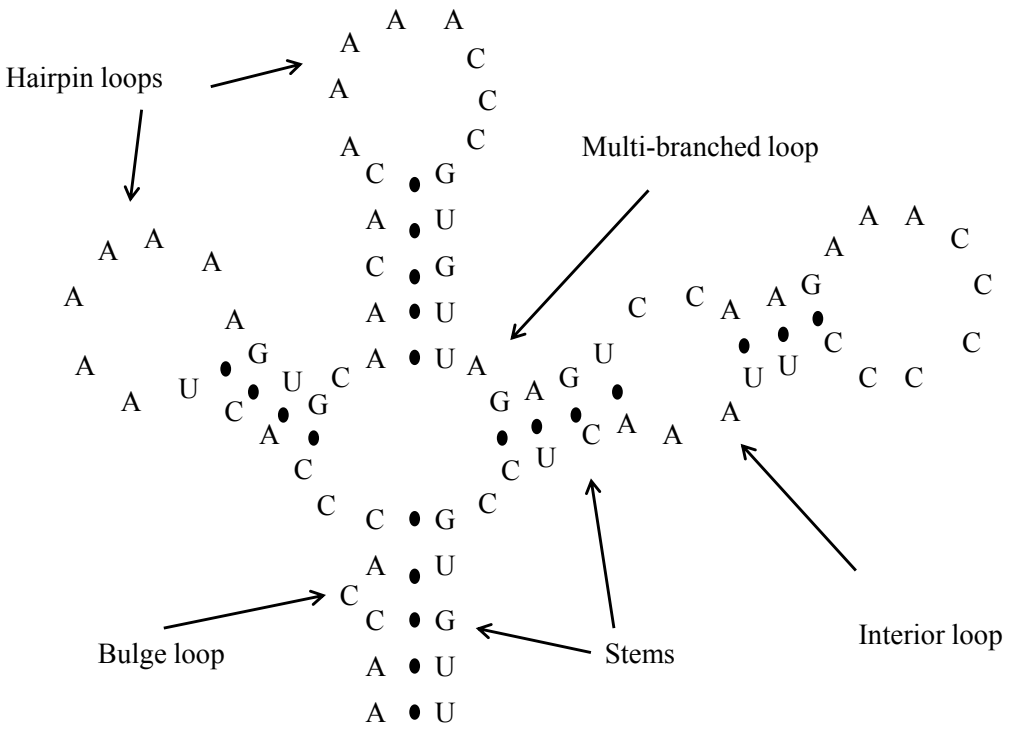


Figure 1.2: General form of ncRNA secondary structure

Transcribed non-coding sequences play diverse and important roles in many biochemical processes. For example, two typical house keeping ncRNAs, tRNA and rRNA, are key components for protein synthesis while MicroRNAs (miRNAs) play critical regulatory roles via interactions with specific target mRNAs in many organisms [23]. Also, short interfering RNAs (siRNAs) are involved in gene silencing in RNAi process [31].

Considering the vital roles ncRNAs play in living organism, any effort which helps know ncRNAs and their biological functions is highly important to modern biology. For example, comparative ncRNA identification, which searches for ncRNAs through evidence of evolutionary conservation, is the state-of-the-art methodology for ncRNA finding. Since the functions of most ncRNAs are determined by both their sequences and secondary structures, comparative ncRNA identification must exploit not only the sequence, but also the structural conservations.

1.2 Our contribution

The origin of context free grammar lies in formal language theory. In biology, stochastic context-free grammar (SCFG) [10] pertains to the problem of alignment and folding homologous families of RNA sequence. A successful application of SCFG is ncRNA classification, which classifies query sequences into annotated ncRNA families such as tRNA, rRNA, riboswitch families. Other secondary structure modeling representations such as base pair probability matrices [18, 41, 45], tree profiles [14, 15], stem graphs [42] etc. have been used in RNA alignment, an important step in novel ncRNA detection. These alignment methods first infer the possible structures of each input sequence and then conduct structural alignment, whose accuracy and efficiently are highly dependent on structural representations. Despite promising output by existing alignment tools, many existing secondary structure representations are highly complicated, incurring high computational cost during alignment. Even with various heuristics or pruning techniques to reduce the time complexity, ncRNA

structural alignment are still more computationally intensive than pure sequence alignment and scale poorly with the number and length of input sequences. Therefore, it remains important to develop an efficient and accurate structural modeling and comparison method.

In this thesis, we design a novel secondary structure representation and show its application in consensus structure derivation through multiple ncRNA alignment and also in ncRNA clustering task. The two contributions are listed below.

First, we design and implement **grammar string**, a novel ncRNA secondary structure representation. A grammar string is defined on a special alphabet constructed from a carefully chosen context free grammar (CFG). It encodes how this CFG generates an ncRNA sequence and its secondary structure. Compared to other secondary structure representations, grammar strings are simple and can take advantage of well-developed algorithms on sequences or strings. For example, grammar strings can convert ncRNA alignment into sequence alignment without losing any structural conservation, rendering highly efficient RNA alignment algorithm. In addition, supporting theories for sequence alignment such as score table design and Karlin-Altual statistics [25] can be applied to grammar string alignment. It is worth mentioning that other string-based secondary structure representations [3, 30, 47] exist. However, those methods focus on deriving ncRNAs' similarities without resorting to alignment and thus cannot be directly applied for consensus structure derivation from homologous ncRNAs.

An interesting application in biology is to classify ncRNAs into families with structural homology. The Rfam database is a standard ncRNA database which will heavily be used during the course of this thesis. This database maintains alignments, consensus secondary structures and such for RNA families. Each family represents a set of RNA sequences that function at the RNA level and share a clear common ancestor. However, there is still a great interest in clustering of these ncRNA families. The reason is Rfam database employs some quality control steps which prevent two families from annotating the same sequences. Therefore, homologous ncRNA sequences might be separated from each other and go to

different families because of restrictions existing in computational steps [12]. Moreover, a single alignment may not capture all the diversity of a group of homologous RNAs. Our second contribution in this thesis is to evaluate utilization of grammar string approach in ncRNA clustering problem. For this purpose, we design a strategy to, first, extract consensus sequence of ncRNA families, and then, generate their corresponding grammar strings based on associated consensus sequence and secondary structure. In the next step, pairwise alignment method is applied to these grammar strings using empirical score table and gap penalties. Finally, ncRNA pairwise alignment scores are fed into a clustering application called MCL to be classified in different clusters.

1.3 Literature review and related works

There are many ncRNAs whose biological functionalities remain unknown to this day. Comparative techniques may help us reveal possible functionalities of novel ncRNAs based on properties of ncRNAs with already known functionalities. The base line is that the ncRNA with similar secondary structures present similar biological functionalities. Existing ncRNA alignment methods can be roughly classified into three basic types.

The first type aligns and folds simultaneously. The most accurate algorithm of this type was developed by Sankoff [36]. However, it is prohibitively expensive with time complexity $O(L^{3N})$ and memory complexity $O(L^{2N})$, where L and N are the length and number of input sequences, respectively. Variants of the Sankoff algorithm have been proposed to reduce the computational time of multiple alignment, such as Stemloc [19], Consan [9], MARNA [38].

The second type of methods first builds a sequence alignment and then folds the alignment [17, 28, 35, 44, 44]. They infer structures from pre-aligned sequences generated using MULTIZ [4], ClustalW [40], or other available sequence alignment programs. The accuracy of these tools is largely affected by the alignment quality. In particular, when homologous ncRNA sequences only share structural similarity, building a meaningful sequence alignment

becomes difficult.

The third type of methods folds input sequences and then conducts structural alignment, yielding higher accuracy. Different tools in this category differ by different secondary structure modeling methods. Although some of them used restricted Sankoff algorithm in their implementations, we classify them into "fold and then align" category because they apply structure prediction in the first step. As our grammar string based alignment belongs to the third category, we discuss related "fold and then align" tools below, focusing on their secondary structure representations.

Several programs encode secondary structure using base pair probability matrices derived from McCastkill's approach [16, 33]. NcRNA alignment is then converted into base pair probability matrix alignment. However, base pair probability matrix comparison is highly resource demanding. For example, pmcomp [18] takes $O(n^4)$ memory and $O(n^6)$ operations for aligning a pair of sequences with length n. More recent implementations such as LocARNA [45] and FOLDALIGNM [41] applied various restrictions or pruning techniques to reduce the time complexity. But they are still much more expensive than sequence alignment.

RNAforester [14, 15] used tree profiles to represent secondary structures. Algorithms on tree alignment are applied for pairwise and multiple alignment computation. The asymptotic efficiency depends on the node number of the tree representation and the maximum degree d of a tree node. For n structures of average size s, their pairwise algorithm has time complexity $O(s^2d^2)$ and space complexity $O(s^2d)$. RNAforester can achieve higher efficiency than base pair probability matrix comparison. However, it is reported [45] that they tend to produce many alignment columns that contain mostly gap characters in the multiple alignment mode. Carnac [42] used stem graphs to represent secondary structures. However, their program cannot accept more than 15 input sequences to remin efficient, limiting its practical usage.

1.4 Road map of this thesis

The remainder of this thesis is organized as follows:

- Chapter 2: Preliminaries and standard definitions of formal language is introduced first. Next, grammar string design is discussed in details.
- Chapter 3: Multiple ncRNA structural alignment experiment designed by our colleague Rujira Achawanantakum [1] is explained in high level. We then focus on those design aspects which pertain to this thesis.
- Chapter 4: An experiment is designed and applied to clustering of ncRNA families using grammar string approach. We will end this chapter with experimental results and conclusions.

Chapter 2

Grammar string design

In this chapter, we first briefly introduce some preliminaries and standard definitions in Formal Languages and Transformational Grammar. We then discuss the use of grammar strings borrowed from formal languages in representation of secondary structure for ncRNAs. Next, a new grammar string will be designed and utilized in a novel representation of these secondary structures. Finally, the associated generating algorithm and the strategy to encode stem structure of ncRNAs will be addressed.

2.1 Preliminaries and standard definitions

2.1.1 Formal Language

We are all familiar with one or more languages and practice at least one of them, e.g., English, French, Italian, etc. in our daily life. Informally, each language consists of a set of symbols and a set of manipulating rules which help us express the facts, feelings, and concepts. However, we need a formal mechanism to study languages mathematically. To begin with, we need to know some terms used in the formal definition of languages.

- A Symbol is an abstract entity having no meaning by itself. The most common symbols are digits, letters from variant alphabets, and special signs such as asterisk.

- An Alphabet is a finite set of symbols. For example, $\{a, b\}$ is an alphabet containing two symbols a and b, and $\{0, 1\}$ is an alphabet with two members 0 and 1.

- A Word or a String is a finite sequence of symbols over an alphabet. For instance, *abab* and 011001 are two words defined on alphabets σ_1 and σ_2, respectively. We use $|.|$ to denote the length of a word, e.g., $|abab| = 4$ and $|011001| = 6$ are the lengths of *abab* and 011001 strings, respectively. The length of a null string is considered 0.

we can now formally define a language using above terms. A language is a set of strings over a set of predefined symbols called alphabet. This set of strings could be an empty, a finite or an infinite one.

2.1.2 Transformational Grammars

Transformational Grammars (Grammars) are used to understand the structure of natural languages and are one of the formal ways of describing a language. Generally, a grammar is defined as a 4-tuple $G = (V, T, P, S)$ where:

- V is a set of symbols called nonterminal, typically S,A,B,...

- T is a set of symbols called terminals, typically 0,1,a,b,...

- P is a set of production rules

- S is the starting nonterminal from V

Each rule in P has the form: $A \rightarrow \alpha$ where A is a variable and α is any concatenation of terminals and nonterminals. If there are more than one candidate in right side of a production rule, the sign | is used to separate them from each other. Moreover, a specific symbol such as ε presents a null string. When a nonterminal is replaced by a null string, no symbol would be added to the string.

As an example, a sample grammar could be defined as $G = (V, T, P, S)$ where $V = \{S, A\}$, $T = \{0, 1\}$ and the production rules from set P are:

$$S \rightarrow 0A|0 \tag{2.1}$$

$$A \rightarrow 10A|\varepsilon \tag{2.2}$$

2.1.3 Derivation process

The sequence of production rules in a grammar that transforms the starting symbol of a grammar into a string out of any nonterminals is called derivation process. The derivation process begins with the starting symbol. In each step, one nonterminal is replaced with one of the possible right hand side production rules. During the whole process, all nonterminals are replaced by terminals and finally the string is produced. The existence of a derivation process proves that a string belongs to a grammar language. In the previous example, the string 0101010 is derived from G using these steps:

$$S \Rightarrow 0A \Rightarrow 010A \Rightarrow 01010A \Rightarrow 0101010A \Rightarrow 0101010A \Rightarrow 0101010 \tag{2.3}$$

As it is seen in (2.1) and (2.3) single-line arrow stands for steps in production rules, whereas double-line arrow shows different steps in a derivation process.

There are two types of derivation: The left-most derivation and the right-most derivation. A left-most derivation is the one in which the left-most nonterminals are always applied in transforming rules. Similarly, a right-most derivation is the one in which the right-most nonterminals are always applied in transforming rule processes. For example, suppose the grammar $G1$ is defined as $G1 = (V, T, P, S)$ where $V = \{S, A, B\}$, $T = \{a, b\}$, and the

production rules in set P are:

$$S \rightarrow AB \tag{2.4}$$

$$A \rightarrow aA|\varepsilon \tag{2.5}$$

$$B \rightarrow bB|\varepsilon \tag{2.6}$$

This grammar defines a language containing empty string, strings of a, strings of b, and strings starting with one or more 'a' and continuing with any number of b. a, b, aaa, $bbbbbb$, aab,and $aaabbbbb$ are some sample words belonging to this language. These strings are generated by production rules in grammar during the derivation process.

Two samples of left-most and right-most derivation steps of the string aab from grammar $G1$ are shown as follows:

$$S \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aabB \Rightarrow aab$$

$$S \Rightarrow AB \Rightarrow AbB \Rightarrow AbbB \Rightarrow Abb \Rightarrow aAbb \Rightarrow abb$$

2.1.4 Parse tree

The derivation process could be presented in the form of a specific tree called parse tree. Parse tree represents the syntactic structure of a string according to some formal grammar. In a parse tree, the root is the starting symbol (S) of the grammar, the interior nodes are labeled by non-terminals of the grammar, and the leaf nodes are labeled by terminals of the grammar. In Figure 2.1 two parse trees $T1$ and $T2$ present the left-most and right-most derivation of the string aab from $G1$.

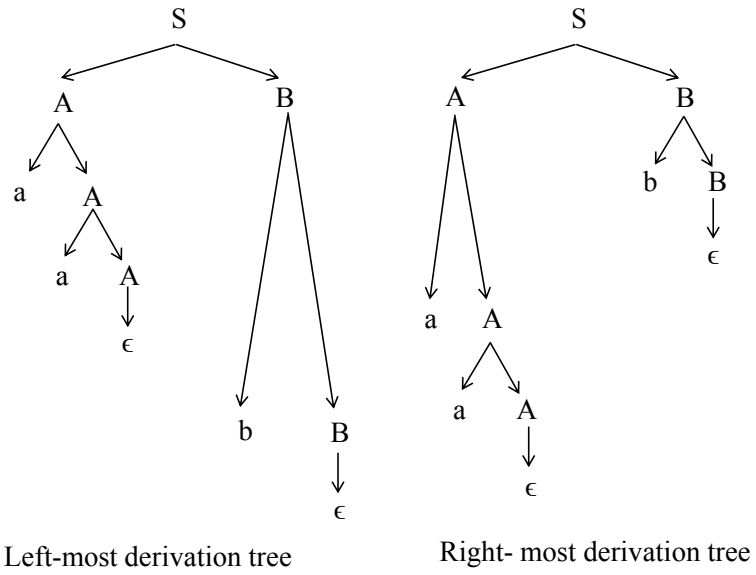


Figure 2.1: "aab" pars trees

2.1.5 ambiguous grammar

A grammar G is an ambiguous grammar if the language generated by G contains a string w which has at least two different parse trees, or equivalently more than one left-most derivation tree. For example, there are two different left-most derivations (Figure 2.2) for generating the string $a + a + a$ using grammar $G2 = (V, T, P, S)$ where $V = \{S\}$, $T = \{a\}$, and with the production rules in set P as:

$$S \to S + S$$

$$S \to a.$$

An ambiguous grammar could be turned into a non-ambiguous one by defining some new

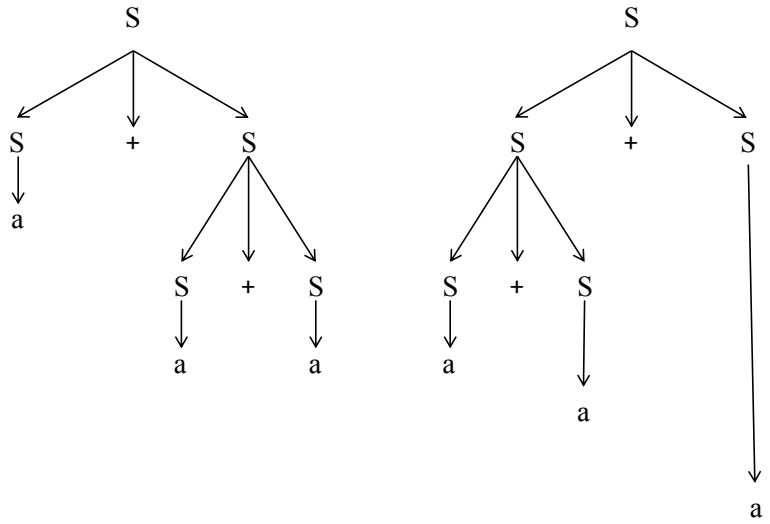


Figure 2.2: Ambiguity in pars tree of "a+a+a" left-most derivation

nonterminals. In example above, we can resolve the ambiguity problem by adding a new nonterminal T to the nonterminal set and modifying the production rules consequently:

$$S \to S + T | F$$

$$T \to F$$

$$F \to a.$$

Figure 2.3 shows the left-most derivation of

$$a + a + a$$

and its corresponding pars tree using non-ambiguous grammar above.

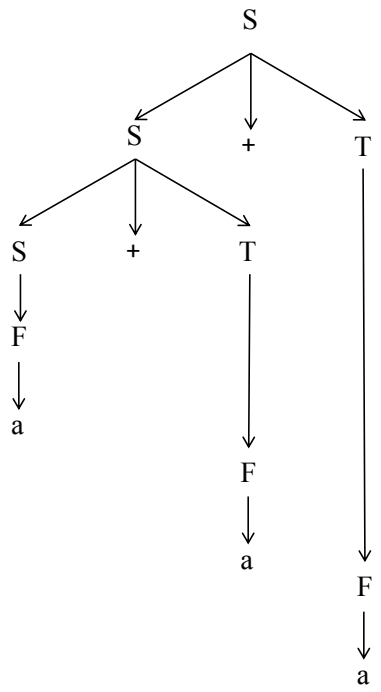


Figure 2.3: Equivalent non-ambiguous pars tree of "a+a+a"

2.1.6 Grammar classes

Chomsky developed a general theory for modeling string of symbols and presented a hierarchy of grammars called **Chomsky hierarchy of transformational grammars** [10]. In other words, Chomsky divided grammars into different classes based on the type and attributes of strings generated by these grammars. Regular grammars, context free grammars, and context sensitive grammars are some examples of these grammar classes.

Regular grammars

A grammar $G = (V, T, P, S)$ is called a regular grammar if all its production rules have the form of $A \to w$ or $A \to wB$, where A and B are variables and w is any combination of terminals including empty string. If a language could be described by a regular grammar

then it is called regular language.

Regular grammars generate strings from left to right or right to left. Another property of regular grammars lies in the fact that the generation of a symbol pair requires to be independent and from two different nonterminals, i.e., regular grammars cannot generate a correlated base pair from a single nonterminal. G introduced in subsection 2.1.2 is such a regular grammar.

Context free grammars

A grammar $G = (V, T, P, S)$ is said to be a context free grammar (CFG) if all productions are of the form $A \rightarrow x$ where A is in V and x is any combination of terminals and nonterminals including null string. The languages generated by context-free grammars are known as the context-free languages.

Context free grammars generate string from outside in. For example, $G3 = (V, T, P, S)$ where $V = \{S\}$, $T = \{a, b\}$, with the production rules in set P as:

$$S \rightarrow aSa|bSb|cSd|aa|bb$$

is an example of a context free grammars. The derivation of "*abaaba*" and "*aacaadaa*" from $G3$ is given, below.

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abaaba$$

$$S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aacSdaa \Rightarrow aacaadaa$$

2.1.7 Context free grammars and ncRNA secondary structure

A typical context free grammar creates strings from outside in and enables to generate strings with nested but non-crossing correlation between pairs of symbols.These attributes make CFG class a proper option to generate **Palindrome** language. Strings of a such language have a symmetric correlation between pair of symbols and are often being read the

same way in either directions left or right (see Figure 2.4). Since secondary structures in nsRNAs are considered as a type of palindrome language, they can be generated by properly designed context free grammars.

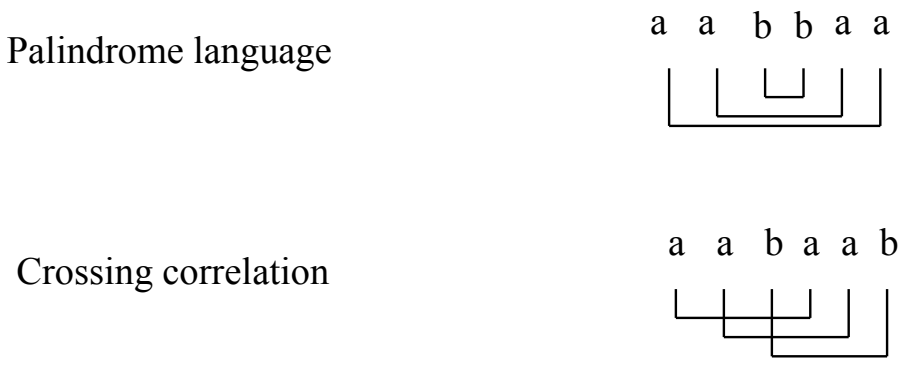


Figure 2.4: Crossing and non-crossing correlation between pairs of symbols

2.2 Grammar string design

Inspired by Jaakkola and Haussler's discriminative classification method [20], we introduce **grammar string**, a representation of an ncRNA sequence in the parameter space of context-free grammar (CFG). Specifically, each ncRNA sequence and its secondary structure are transformed into a string defined on a new alphabet, where each character corresponds to a production rule in a CFG.

We first introduce an unambiguous CFG for ncRNA sequence generation. Using the

chosen CFG as an example, we formally define grammar strings for modeling an ncRNA sequence and its secondary structure.

2.2.1 An unambiguous CFG for ncRNA generation

NcRNA structures without pseudo-knots can be derived by CFGs [10]. As mentioned in previous section, a CFG is defined by a set of nonterminals, a set of terminals, a start nonterminal, and a set of production rules of the form $V \rightarrow \alpha$. V is a single nonterminal symbol, and α is a string of terminals and/or nonterminals. By recursively replacing nonterminals on the right hand side of each production rule, an ncRNA sequence and its secondary structure can be derived from a CFG. In this work, all our ncRNA sequences and their structures will be generated from G4, a light-weight CFG introduced by Dowell and Eddy [8], using leftmost derivation. Following the general definition of a CFG, $G4$ has a finite set of nonterminal $V = \{\mathcal{S}, \mathcal{T}\}$, a finite set of terminal symbols $T = \{A, C, G, U, \varepsilon\}$, and a finite set of production rules defined as below:

- $\mathcal{S} \rightarrow a\mathcal{S}|\mathcal{T}|\varepsilon$
- $\mathcal{T} \rightarrow \mathcal{T}a|a\mathcal{S}\hat{a}|\mathcal{T}a\mathcal{S}\hat{a}$

where $a \in \{A, C, G, U\}$ and $\hat{a} \in \{A, C, G, U\}$. a and $\hat{a}$ form complementary base pairs such as A-U and G-C. In order to generate the unstructured single strand 'C' at 3' end and the two outmost base pairs in sequence tRNA_1 in Figure 2.7, the following production rules from G4 are called: $\mathcal{S} \rightarrow \mathcal{T}$, $\mathcal{T} \rightarrow \mathcal{T}C$, $\mathcal{T} \rightarrow G\mathcal{S}C$, $\mathcal{S} \rightarrow \mathcal{T}$, $\mathcal{T} \rightarrow U\mathcal{S}A$. Continuing to replace $\mathcal{S}$ by correctly chosen production rules, we can derive tRNA_1. The sequence of production rules used for ncRNA structure generation is called a **derivation**.

Using the leftmost derivation, an **unambiguous** CFG can guaranteer a **unique** derivation for a given ncRNA sequence and its secondary structure. For example, by using the unambiguous grammar G4, we have only one choice when choosing a production rule to derive a sample ncRNA (Figure 2.5) secondary structure in Figure 2.6. For a more de-

tailed introduction about unambiguous CFGs, we refer readers to the review by Dowell and Eddy [8], where several light-weight unambiguous CFGs including $G4$ are discussed.

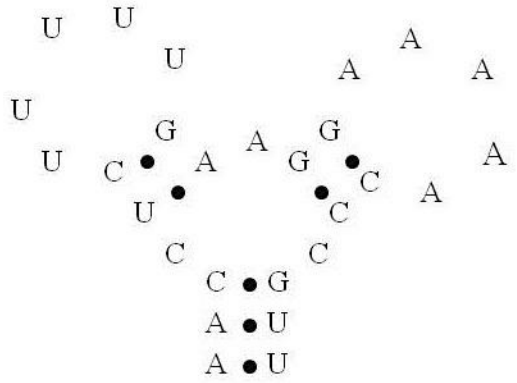


Figure 2.5: Secondary structure of a sample ncRNA

2.2.2 Grammar string generation algorithm

Each ncRNA secondary structure has a unique leftmost derivation from an unambiguous CFG, producing a one-to-one mapping between a structure and a production rule sequence. Intuitively, homologous ncRNAs with similar structures will share similar derivations. This motivates us to represent an ncRNA sequence and its secondary structure in the parameter space of a CFG. Thus, ncRNA structural comparison is converted to the comparison of their derivations.

In order to represent an ncRNA structure using its derivation, we introduce a new alphabet, where each character corresponds to a production rule in a CFG. One example alphabet derived from $G4$ is defined below.

- Use upper case character of a to represent production rule $\mathcal{S} \rightarrow a\mathcal{S}$. For example, use A to represent $\mathcal{S} \rightarrow A\mathcal{S}$.
- Use $|$ to represent $\mathcal{S} \rightarrow \varepsilon$.

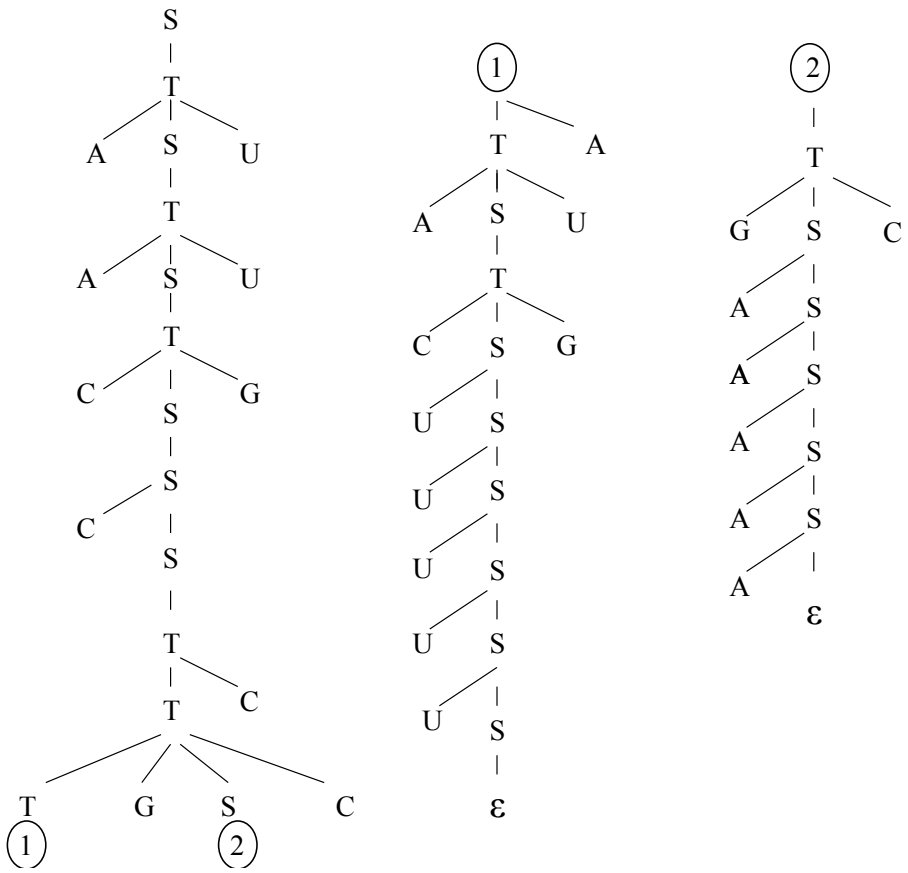
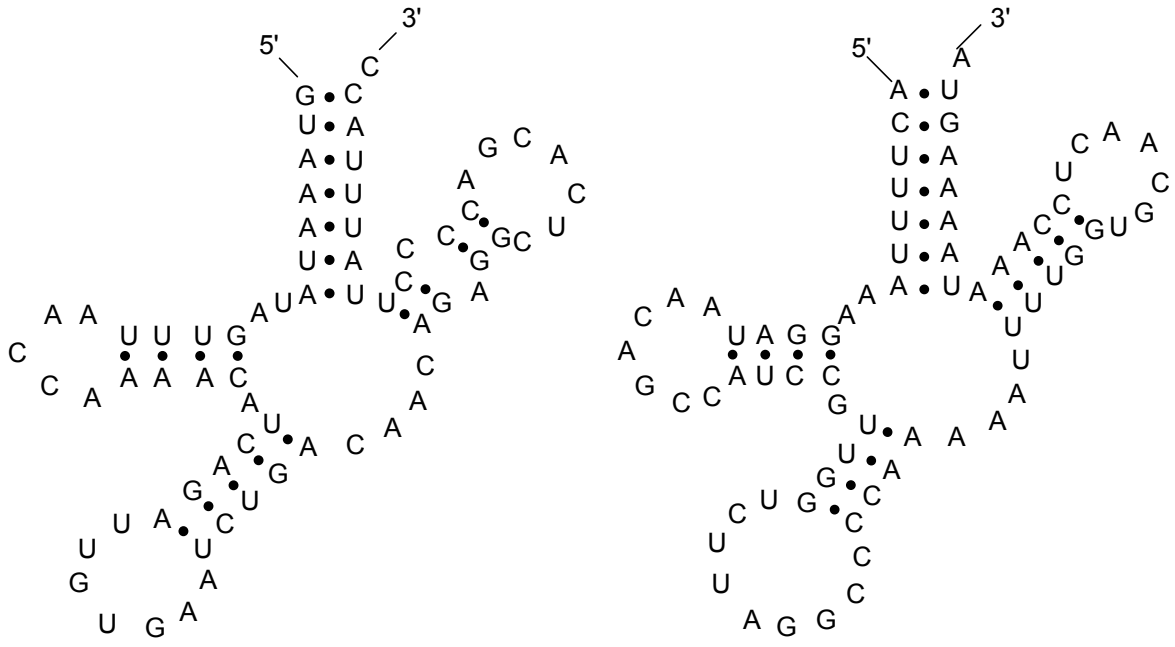


Figure 2.6: Left most derivation pars tree of Figure 2.5

- Use lower case character of a to represent production rule $\mathcal{T} \to \mathcal{T}a$. For example, use c to represent $\mathcal{T} \to \mathcal{T}C$.
- Use P to represent base pair emission $\mathcal{T} \to a\mathcal{S}\hat{a}$.
- Use a special character # to indicate branching $\mathcal{T} \to \mathcal{T}a\mathcal{S}\hat{a}$.
- No character is needed for production rule $\mathcal{S} \to \mathcal{T}$.

Thus, the new alphabet is $\mathcal{A} = \{A, C, G, U, a, c, g, u, P, —, \#\}$. If these production rules are used on DNA sequences, we can simply replace $U(u)$ with $T(t)$. For brevity, we name a string defined on the above alphabet a **grammar string**. As an example, the derivation for generating the unstructured single strand 'C' at 3' end and the two outmost base pairs in sequence tRNA_1 of Figure 2.7 is: $\mathcal{S} \to \mathcal{T}$, $\mathcal{T} \to \mathcal{T}$ C, $\mathcal{T} \to$ G $\mathcal{S}$ C, $\mathcal{S} \to \mathcal{T}$, $\mathcal{T} \to$ U $\mathcal{S}$ A.

Thus, the corresponding grammar string is cPP using the alphabet $\mathcal{A}$. Note that we don't distinguish different base pairs (i.e. A-U, G-C, and G-U if allowed) in a grammar string. All base pairs are represented as P in order to maximize the alignment score between homologous ncRNAs that share high structural similarity but low sequence similarity. Figure 2.7 shows the utility of grammar strings in detecting structural similarity between two tRNA sequences from the human genome. Because of low sequence similarity, BLAST [2] fails to align them. However, their structural similarity yields a meaningful global alignment between their corresponding grammar strings with 69% identity. In theory, our grammar string




```
tRNA_1: GUAAAUAUAGUUUAACCAAAACAUCAGAUUGUGAAUCUGACAACAGAGGCUCACGACCCCUUAUUUACC
        (((((((..((((.....)))).(((((.......))))).....((.(.......)).)))))))).
                              Grammar string
        cPPPPPPPUA#caac#aPPPPAACCA|PPPPPUUGUGAA|PPAcPPCUCACGA|

tRNA_2: ACUUUUAAAGGAUAACAGCCAUCCGUUGGUCUUAGGCCCCAAAAAUUUUGGUGCAACUCCAAAUAAAAGUA
        (((((((..((((.......)))).((((.........)))).....(((((.......)))))))))))).
                              Grammar string
        aPPPPPPPAA#uaaa#gPPPPAACAGCC|PPPPUCUUAGGCC|PPPPPUGCAACU|

                              Pairwise alignment
        cPPPPPPPUA#aac#aPPPPAAC--CA|PPPPP-UUGUGAA|PPPPPCUCACGA|
        aPPPPPPPAA#aaa#gPPPPAACAGCC|PPPPPCUUAGGC-|PPPPPUGCAACU|
         ******* **** * *******  * ****** **  *  ******  **   *
```

Figure 2.7: Two tRNA sequences from the human genome and the alignment of their grammar strings.The stars below the alignment denote exact matches.

```
void parse(i, j)
{
if i >= j
        print '|';
        return;
else if X_i is a single stranded base
        print uppercase of X_i;
        i++;
        parse(i,j);
else if  X_j is a single stranded base
        print lowercase of  X_j;
        j--;
        parse(i,j);
else if  X_i and X_j form a base pair
        print 'P';
        i++ and j--;
        parse(i,j);
else
        print '#';
        k = the position that forms a base pair with X_j;
        parse(i,k-1);
        parse(k,j);
}
```

Figure 2.8: Algorithm for generating a grammar string for substring $X_{i..j}$.

generation process consists of two steps. First, write the production rule sequence for an ncRNA sequence and its secondary structure. Second, transform the sequence of production rules into a grammar string according to the definition of grammar string alphabet. In practice, we use an efficient dynamic programming algorithm to design a grammar string for an ncRNA structure directly, skipping the step of parsing an ncRNA sequence using a CFG. The algorithm has time complexity $O(L^2)$, where L is the length of the ncRNA sequence.

Let X be an ncRNA sequence with its predicted or annotated secondary structure. i and j are indexes in X. X_i is the base at position i. Figure 2.8 sketches the dynamic programming algorithm generating a grammar string for substring $X_{i..j}$. In order to generate the complete grammar string for sequence X, one should call parse(1, L).

2.2.3 Design a new grammar string of Rfam database

In the grammar string presented in Section 2.2, all base pairs are mapped to the same alphabet P. Therefore, all base pairs are considered the same in a grammar string. Here instead we design a new grammar string which enables us to distinguish between different types of complementary base pairs. Our goal is to compare the performance of a grammar string when we feed it with more details about base pairs nucleotides and their decoration.

To design a new grammar string we still use $G4$ context free grammar, but expand its set of alphabets from $A = \{A, U, C, G, a, u, c, g, P, \#, —\}$ to

$$A = \{A, U, C, G, a, u, c, g, <, >, [,], (,), E, F, I, J, L, O, Z, \#, —\}.$$

As seen, instead of P which is a single representative for base pairs in our previous grammar string design, some other new alphabets are added to the new alphabet set. Each new alphabet is a representative for a specific type of base pair. In Table 2.1, it is seen how base pairs are assigned to grammar string alphabets. For more details, read Section 4.2.1. Both the previous and new generated consensus sequences and grammar strings of family RF00756 along with the corresponding secondary structure are shown below, respectively.

```
AAAAAAAA.AAAAAAAAAAAAAAAAAAAAUUUGAAUUUUUUUUUUUUUUUUUUUUUUUUUUUU,RF00756
<<<<<<<<.<<<<<<<<<<<<<<<<<<<<.......>>>>>>>>>>>>>>>>>>>>>>>>>>>>
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPUUUGAAU|


AAGAAGUG.GUUUACCGUCCCACAUACAUUUUGAAUAUGUAUGUGGGACGGUAAACCGUUUCUU,RF00756
<<<<<<<<.<<<<<<<<<<<<<<<<<<<<.......>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<]<<()]]>>><[[]>[[[<[<><[<>UUUGAAU|
```

Table 2.1: Observed $\mapsto$ Replacement

Observed	Replacement
AU	$<$
UA	$>$
CG	[
GC	]
GU	(
UG	)
EE	E
FF	F
II	I
JJ	J
LL	L
OO	O
ZZ	Z

2.2.4 Grammar pattern for encoding stem structures

The number of stems and their relationship largely define the basic "shape" of a secondary structure. For example, the cloverleaf structure of a tRNA sequence consists of four stems: acceptor stem, D stem, anticodon stem, and TΨCG stem. The precursor structure of a miRNA usually contains only one stem. According to the definition of grammar strings, three characters $P, \#$, and | encode the number and relative positions of all stems in an ncRNA secondary structure. If we simply remove all single stranded regions (i.e. substrings only consisting of A, C, G, U, a, c, g, u) from a grammar string, we can use a simplified grammar string to represent the abstract stem structure for an ncRNA sequence. For brevity, we name a simplified grammar string a **grammar pattern**, which is a string defined on a reduced alphabet $\{P, \#, |\}$. A grammar string can be converted into a grammar pattern in two steps: 1) remove all substrings representing single stranded regions, and 2) reduce every substring consisting of only Ps as a single P. Thus, the grammar pattern for sequence tRNA_1 in Figure 2.7 is $P\#\#P|P|P|$, where each P denotes a stem. There are four Ps, denoting four stems. The end of each stem is marked by |. Number of # defines the number of bifurcations.

Different distributions of the same number of stems can yield highly different secondary structures. Figure 2.9 shows how grammar patterns can account for different structures with

the same number of stems. Note that all these grammar patterns are generated using G4 as the chosen CFG. If other unambiguous CFGs are used to generate grammar strings for the same structures, different sets of grammar patterns might be produced. Ignoring all

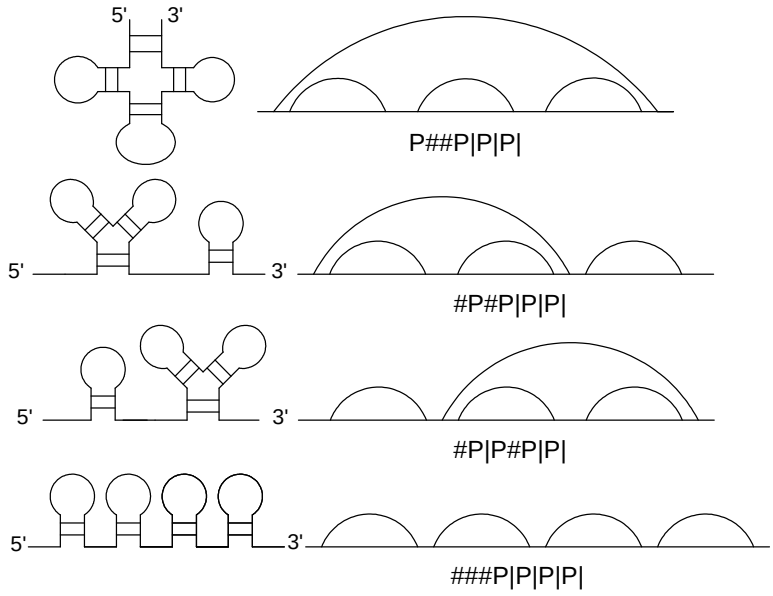


Figure 2.9: Four different stem structures and their grammar patterns. The left column shows the 2D representation of an ncRNA folding. The right column shows the distributions of stems along an ncRNA sequence. All grammar patterns are generated using G4 (our chosen unambiguous context-free grammar).

single stranded regions and length of each stem, grammar patterns only provide a coarse-grained description of ncRNA secondary structures. However, because of the high efficiency of pattern matching, grammar patterns can be used to speed up grammar string comparison. For example, we do not expect significant structural similarities between a tRNA and a miRNA sequence. Instead of using Needleman-Wunsch [37] like alignment algorithm between their grammar strings, a constant time grammar pattern matching program can be applied as a filtration step. This filtration is particularly important when we aim to derive the consensus structure of multiple putatively homologous ncRNAs. Although these sequences are expected to be sequenced from the same gene family, it is possible that some of the sequences are from other regions. Thus, we can use the grammar pattern matching technique to exclude contaminated sequences, ensuring a multiple sequence alignment with good quality. The

same technique can be used to remove possible errors introduced by MFE-based secondary structure prediction tools.

2.2.5 Other grammar pattern design strategies

Designed grammar pattern in Section 2.2.4 encodes the stem structure of an ncRNA sequence. However, more complicated grammar patterns can be designed to carry information about both sequence and secondary structure of a sample ncRNA. Following designed strategies are such examples.

- **Strategy (A)**

  1. Preserving all bifurcation (#) and end (|) grammar alphabets.
  2. Removing individual bases from grammar string.
  3. Keeping only one P as the representative of base pairs if any base pair is seen between bifurcation and end alphabets.

- **Strategy (B)**

  1. Preserving all bifurcation (#) and end (|) grammar alphabets.
  2. Keeping one P as a representative for each segment of base pairs.

- **Strategy (C)**

  1. Preserving all bifurcation and end grammar alphabets.
  2. Finding the total number of Ps, the total number of left derived individual bases (capital alphabets), and the total number of right derived individual bases (small alphabets) existing in the grammar string.
  3. Finding the total number of segments in which base pairs (Ps) occur, the total number of segments in which left derived individual bases occur, and the total number of segments in which right derived individual bases occur.

4. Calculating average length of three segments (found in the previous step) by dividing the total number of the corresponding grammar string alphabet to the number of segments in which the alphabet occur.
5. Calculating the average of tree averages (calculated in the previous step) and calling it "total average".
6. Processing the grammar string again from left to right.
7. Preserving all bifurcation (#) and end (|) grammar alphabets.
8. Putting one P, A, or a for each observed segment if the number of characters in that segment is greater than or equal to an empirical threshold $\times$ total average.

Chapter 3

Multiple ncRNA structural alignment based on grammar string

In this chapter, we present a brief overview on multiple alignment of ncRNAs secondary structure. We must mention that the main effort in leading this research project has been done by our colleague Rujira Achawanantakun [1]. Although the final goal of this project, accomplished by Rujira Achawanantakun, has been to derive consensus structure through multiple ncRNA alignment, in this thesis we only focus on those parts of the research which have been done by the author of the current thesis. These include creating score table for grammar strings pairwise alignment and extracting consensus sequence of ncRNA families. We refer interested readers to [1] for more details on other steps of this experiment as well as the final result of this research work.

3.1 Summary of designed experiment for multiple ncRNA structural alignment by Achawanantakun [1]

Multiple ncRNA alignment has important applications in homologous ncRNA consensus structure derivation, novel ncRNA identification, and known ncRNA classification. As many

ncRNAs' functions are determined by both their sequences and secondary structures, accurate ncRNA alignment algorithms must maximize both sequence and structural similarity simultaneously, incurring high computational cost. Faster secondary structure modeling and alignment methods using trees, graphs, probability matrices have thus been developed. Despite promising results from existing ncRNA alignment tools, there is a need for more efficient and accurate ncRNA secondary structure modeling and alignment methods. To this end, Achawanantakun designed an experiment to derive consensus structure through multiple ncRNA alignment. Major steps of aligning multiple ncRNA sequences experiment are sketched below:

1. Use an ab initio secondary structure prediction tool to predict both the optimal and sub-optimal secondary structures of each input sequence based on minimum free energy criteria.

2. Generate a grammar string for each predicted secondary structure. If an ncRNA sequence has more than one structure predicted, multiple grammar strings will be generated.

3. Transform each grammar string into a grammar pattern. Use a voting mechanism to choose the most popular grammar pattern that mostly likely represents the native stem structure shared by the input sequences. All grammar strings that are not consistent with the chosen grammar pattern will be discarded.

4. Apply a progressive multiple sequence alignment method on remaining grammar strings.

5. Derive the consensus secondary structure from multiple grammar string alignment. Transform grammar string alignment into ncRNA sequence alignment using the ncRNA sequences and their predicted structures as references.

In Chapter 2, we introduced an approach to generate grammar string from an ncRNA sequence and its secondary structure (Step2). We also addressed, in details, how to extract

grammar pattern from a grammar string (Step 3). In the next two sections, we first explain how to create a score table for grammar string alignment and then how to extract consensus sequence of ncRNA families to generate their corresponding grammar string.

3.2 Score table design for grammar string alignment

Pairwise alignment is a fundamental step to multiple alignment and clustering. Existing alignment algorithms such as Needleman-Wunsch [37] can be directly applied to grammar strings when a score table defined on grammar strings' alphabet is imported. Following the common practice in score table design, we use maximum-likelihood ratio to derive the score between every pair of characters in grammar strings' alphabet $\mathcal{A}$. For each pair of characters a, b in $\mathcal{A}$, the score between a, b is $s(a, b) = \log \frac{\Pr(a,b)}{\Pr^0(a,b)}$. $\Pr(a, b)$ is the target probability of a, b in a set of true alignments and $Pr^0(a, b)$ is the background probability that a and b are aligned. Assuming that a and b are independent in the background model, we get $\Pr^0(a, b) = Pr^0(a) \times Pr^0(b)$. Because ncRNA family database Rfam [13] provides a large number of annotated ncRNA sequences, their alignments, and their associated secondary structures, we obtain both the target and the background probabilities from Rfam. In summary, we present following steps of designing a score table for grammar string alignment.

1. Build an alignment training set by randomly picking a large number of pairwise ncRNA alignment from Rfam 9.1's seed alignments. Some criteria are applied to select alignments with reasonably high quality. For example, if a pairwise alignment contains too many gaps, it will not be included in the training set. After applying the selection criteria, we had 18487 pairwise alignments in the training set.

2. Transform each pair of ncRNA sequence alignment into an alignment between grammar strings using the given secondary structure annotations by Rfam.

3. Compute the target probability $Pr(a, b)$ for each pair of aligned characters a, b in the above grammar string alignments.

4. Generate grammar strings for a large number of ncRNA sequences that are randomly picked from full families of Rfam 9.1. Compute the background probabilities $Pr^0(a)$ and $Pr^0(b)$ from these grammar strings.

The complete score table for grammar string alignment can be found at our website (Appendix B). All exact matches have big positive scores. And bifurcation starting character # and stem ending character | can only be aligned with themselves or cause a gap. This is consistent with our intuition because it is not meaningful to align a bifurcation character with a base pair or a single stranded base.

Insertions or deletions of P or single stranded characters correspond to insertions or deletions of a base pair or single stranded bases in the ncRNA sequence alignment. Empirical experiments are conducted to choose default values for their gap opening and extension costs. The default gap opening score is slightly smaller than the lowest number in the grammar string's score table. The default gap extension cost is set as 1/10 of the opening cost. We assign bigger gap penalties for structural characters # and | in order to force corresponding stems or single stranded regions to be aligned together.

3.3 Extracting consensus sequence of ncRNA families

We select a data set which includes 452 randomly chosen families from BRAliBase 2.1, an enhanced RNA alignment benchmark [46]. This data set contains a diverse set of ncRNA families with different average sequence identity, length, and structural conservation.

In order to generate grammar string of each family, we first need to extract its corresponding consensus sequence. For this purpose, we design an empirical strategy to extract the consensus sequence of an Rfam family from its multiple sequence alignment and consensus secondary structure. We propose different approaches to generate non-complementary bases and complementary base-pairs of consensus sequence.

To extract consensus individual base of a specific column in a multiple alignment required

steps are as follows:

1. Count the total number of different nucleotides occurred in this specific column of all sequences. These nucleotides might be any of A, U, C, G nucleotides and in any number plus dot character (".") which is representative of gap in multiple alignments.

2. If the number of counted dots in the column is equal or larger than a fraction of the total number of sequences, we put "." in that column of consensus sequence. Otherwise, we sort non-dot nucleotides in descending order based on the number of occurrence and then pick them from maximum in descending direction until the sum of selected nucleotides satisfies an empirical threshold. Finally, based on Tables 3.1, 3.2, and 3.3, a proper alphabet is selected to put in the corresponding column of consensus sequence.

Table 3.1: Observed $\mapsto$ Replacement

Observed	Replacement
A	A
U	U
C	C
G	G

Table 3.2: Observed $\mapsto$ Replacement

Observed	Replacement
anagrams of (UA)	W
anagrams of (AC)	M
anagrams of (AG)	R
anagrams of (UC)	Y
anagrams of (UG)	K
anagrams of (CG)	S

Table 3.3: Observed $\mapsto$ Replacement

Observed	Replacement
anagrams of (AUC)	H
anagrams of (AUG)	D
anagrams of (ACG)	V
anagrams of (UCG)	B

If none of these cases happen, put N in that column. Anagrams in these tables stand for all possible rearrangements of letters in the candidate nucleotides, e.g., AU has two anagrams: AU and UA.

By mapping the observed nucleotides (left column) to our defined alphabets (right column), we are trying to reveal the degree of conservation in that column. For example, when the number of an observed nucleotide is high enough to satisfy the empirical threshold on its own, we map that nucleotide to itself and it implies that column is well conserved. However, when the summation of more than one nucleotide is needed to satisfy the threshold, we map them to our defined alphabets so as to show the degree of conservation in that column. Obviously, the more selected nucleotides, the less degree of conservation in that column.

To extract base pairs of a consensus sequence, we simply put AU in associated columns of that sequence. This is because in our designed grammar string all complementary base pair types are considered the same and are mapped to a single grammar alphabet P.

Remark 3.1. I*n the designed alignment pipeline, multiple structures are allowed to be predicted for each input sequence. As a result, multiple grammar strings will exist for a single ncRNA sequence. However, predicted structures for the same ncRNA sequence can differ significantly. It is important to align only structures that are likely to be consistent with the native structure of the homologous sequences. For this purpose, grammar pattern introduced in grammar string chapter is applied to pre-select grammar strings for multiple alignment.*

Chapter 4

Clustering based on grammar string

Cluster analysis or clustering is generally the task of assigning objects of a set to separate groups called clusters. Objects in the same cluster are more similar (in some sense) to each other than to those in other clusters. This technique is vastly used in different areas of bioinformatics. More specifically, a group of ncRNAs are classified into clusters, called families, based on their sequential and structural homology. However, as mentioned in Section 1.2, restrictions existing in computational steps of generating ncRNA families might separate some of homologous ncRNA sequences from each other by assigning them to different families. Therefore, designing techniques that enable us to identify this type of homologos but separated families is still an interesting research area in bioinformatics.

In this chapter, we utilize grammar strings in ncRNA classification (excluding pseudoknots) and find clusters in a particular database, Rfam. Organization of this chapter is as follows: Section 4.1 defines ncRNA classification problem and the big picture of our methodology to tackle this problem. Section 4.2 discusses steps of designed experiment in more details. Section 4.3 presents experimental results of our approach.

4.1 NcRNA clustering; problem definition and approach

A. Problem definition

In Rfam database, non-coding RNAs are classified into different families based on sequence alignments and statistical profiles known as covariance models. Examples of such families are tRNAs, microRNAs, and such. Each family represents a set of RNA sequences that, first, can be reasonably aligned, and second, function at the RNA level and share a clear common ancestor. These two criteria, however, impose fundamental limitation on finding homologous families of RNAs in Rfam.

The problem with Rfam database comes from the fact that there may be two homologous families with a common ancestor (same functionality) but too divergent to be aligned, or with acceptable alignment but distinct functionality. To overcome these issues, the clan concept is borrowed from the MEROPS and Pfam databases [6,11] to take an alternative approach to ncRNA families classification. This approach still get the right balance of sequence sensitivity and specificity when it is applied to ncRNA families. Here, some of the internal quality control measures used by Rfam can be relaxed for the clanned families. These clans describe explicit relationships between families that either clearly share a common ancestor but are too divergent to be reasonably aligned, or groups of families that could be aligned, but have clearly distinct functions.

As an example, CL0002 clan contains five homologous families RF00030, RF01577, RF00373, RF00010, RF00011 and RF0009. These families contain ribozyme RNAs involved in processing of pre-tRNA and pre-rRNA sequences. However, they are difficult to align to each other. Furthermore, RF01577 and RF00030 are distinct molecules from a functionality point of view [21].

Another clan of interest is CL00011. This clan contains RF00128 and RF00083 Rfam families which are homologous, but distinct in functionality. Small RNAs sequences in RF00128 activate expression of RF00083 while those in RF00083 regulate the translation of a coding gene [22].

B. Our methodology towards ncRNA classification process

Our goal is to evaluate the possibility of using grammar string approach to identify existing clans in Rfam database, and also to discover novel clan families. In other words, we would like to apply a clustering method to derived grammar strings and evaluate the performance of grammar string approach in ncRNA classification problem.

In our proposed approach, we take the following steps:

i) Designing and applying two consensus sequence derivation strategies to Rfam families.

ii) Generating two different corresponding grammar strings $GS1$ and $GS2$ based on the associated consensus sequence and secondary structure.

iii) Extracting two subsets of grammar strings from each set of $GS1$ and $GS2$ as data sets for our experiment.

iv) Applying pairwise alignment method to all the subsets, built in previous step, using empirical score table and gap penalties.

v) Feeding ncRNA pairwise alignment scores into a clustering application called MCL (Markov cluster Algorithm) to be classified into different clusters.

In next section, we discuss these steps in more details.

4.2 Step by step design for clustering experiment

4.2.1 Extracting consensus sequences in Rfam families

In order to generate grammar strings in Rfam families, we first need to extract their consensus sequences. We design two empirical strategies to extract the consensus sequence of an Rfam family from its multiple sequence alignment and consensus secondary structure. In one strategy, we assume all base pairs are the same, whereas in the other one, different base pairs are separated from each other.

To extract consensus sequences of Rfam families without distinguishing between base pairs, we apply the same strategy introduced in Section 3.3, and therefore, the approach will not be repeated here again. To extract complementary base-pairs of consensus sequences with distinguished base pairs from multiple alignment of an Rfam family, a new strategy is designed and presented here.

Extracting base pairs of a consensus sequence from multiple alignment

In this case, the number of different occurrences of base pairs are counted. Possible base pairs could be any of AU, UA, CG, GC, GU, UG, or even two dots to present un-conserved base pairs in multiple alignment. If the number of occurring dot base pairs is larger than or equal to an empirical fraction of family members, we put dot in the corresponding columns of consensus sequence and secondary structure. Otherwise, we select two most occurring base pairs. Denoting the first and second most repeated base pairs and number of non-dot base pairs by X, Y, and N, respectively, and assuming three thresholds $th_1 = 0.6, th_2 = 0.45$, and $th_3 = 0.4$, we arrive at five different cases to be handled as follows:

1. If (the count of X $\geq th_1 \times N$) then we put X in the consensus sequence (e.g., X=AU then we put AU in corresponding columns of consensus sequence).

2. If ($th_2 \times N \leq$ the count of X $< th_1 \times N$) and (the count of Y $< th_3 \times N$) then we still put X in the consensus sequence (e.g., X=AU then we put AU in corresponding columns of consensus sequence).

3. If ($th_2 \times N \leq$ the count of X $< th_1 \times N$) and (the count of Y $\geq th_3 \times N$) then we use the the Figure 4.1 to set the corresponding columns in the consensus sequence.

4. If (the count of X $< th_2 \times N$) and (the count of Y $< th_3 \times N$) then we put ZZ in the corresponding columns of the consensus sequence.

Observed	Replacement
AU	EE
UA	FF
CG	II
GC	JJ
GU	LL
UG	OO

Figure 4.1: Mapping in the consensus sequence

4.2.2 Strategy of extracting data sets for our experiment

In Subsection 4.2.1, we explained how $C1$ and $C2$ as two sets of consensus sequences are built from Rfam families. In $C1$, all base pairs are shown by single alphabet 'P' while in $C2$ base pairs are distinguished based on the type of nucleotides and their decoration. By adding conserved secondary structures of Rfam families to $C1$ and $C2$, we can apply two grammar string approaches to these data sets. The version of grammar string design without distinguishing between base pairs will be applied to $C1$, whereas the one with separated grammar string alphabets for each type of complementary base pair will be applied to $C2$. As a result, two sets $GS1$ and $GS2$ of grammar strings will be generated for whole Rfam families.

So far, we have designed two grammar string approaches in Sections 2.2.1 and 2.2.3. From

now on, we refer to these by "old" and "new" labels, respectively. To prepare the data sets for the experiment, we first need to generate two grammar sets of whole Rfam families based on our old and new grammar string designs. To this end, we extract secondary structures of Rfam families from Rfam database and associate them with their corresponding consensus sequences in $C1$ and $C2$ and create $CS1$ and $CS2$ data sets, respectively. Now, we apply our old grammar string approach to $CS1$ and generate grammar string set $GS1$, and apply the new grammar string approach to $CS2$ and generate grammar string set $GS2$. In next two subsections, we will describe how to prepare data sets for our experiment from $GS1$ and $GS2$.

Strategy for extracting Rfam families of clan database

To study the performance of two different grammar string designs in identifying well-known clans, they need to be tested on Rfam families whose clusters have already been identified. Since generally grammar string approach is a representation for ncRNAs secondary structure, we would like to see if this approach is able to identify and consequently cluster clan family members with similar secondary structures but different functionalities. For this purpose, we take advantage of grammar pattern strategy. We define label "conserved" for clan families whose Rfam family members have the same grammar pattern. By applying tree strategies in Section 2.2.5, we extract fourteen conserved clans. Table 4.3, shows these conserved clans, their family members as well as their corresponding grammar patterns. we now extract grammar strings of conserved clans from $GS1$ and $GS2$ and name them old and new grammar strings of conserved clans, respectively.

Strategy for extracting Rfam families which are not in clan database

To study the performance of two different grammar string designs in discovery of novel ncRNA clusters, grammar strings of Rfam families whose clusters have not been discovered yet are required. We extract this subset of grammar strings from both $GS1$ and $GS2$ sets

and call them old and new grammar strings of Rfam−Clan, respectively. Here and hereafter, we use "Rfam−Clan" grammar strings to denote the entire grammar strings of Rfam families excluding members of Clan database.

4.2.3 Pairwise alignment

After extracting four data sets (old/new conserved clans and old/new Rfam−clan grammar strings), we run pairwise alignment application on these data sets. More specifically, we design two empirical score tables called as "old" and "new" score tables. In the old score table, which will be used in pairwise alignment of old grammar strings, all values are chosen empirically, following a set of guidelines given below.

- Individual bases do not align with base pairs, bifurcations (#), or end of derivations (|).
- Left derivation individual bases do not align with right derivation individual bases and viceversa.
- Base pairs do not align with bifurcations (#) or end of derivations (|).
- Bifurcations (#) do not align with end of derivations and viceversa.
- The score of aligning two same alphabets is positive and that of aligning two different alphabets is negative.
- The score of aligning two bifurcations (#), or two end of derivations (|) is greater than aligning two other equivalent grammar string alphabets.
- Individual bases $\{A, G\}$ and $\{C, U\}$ are aligned better than other combinations of individual bases. Therefore, alignment score of these individual bases is still negative, but greater than alignment score of other individual bases.

To construct the new score table which is used for pairwise alignment of the new grammar strings, we take advantage of RIBOSUM matrices (see Appendix A). These matrices are designed by Robert.J Klein and Sean.R Eddy [27] to search for homologs of a single RNA molecule in a sequence database, based on secondary structure. There are the guidelines to build the new score table:

- The left derivation individual base columns existing in a left derivation individual base row are filled in with scores from the corresponding table of RIBOSUM matrix. The remaining columns in such a row (a left derivation individual base row) will be filled in with a very large negative number so as to avoid selecting alignments associated with those columns.

- The right derivation individual base columns existing in a right derivation individual base row are filled in with scores from the corresponding table of RIBOSUM matrix. The remaining columns in such a row (a right derivation individual base row) will be filled in with a very large negative number so as to avoid selecting alignments associated with those columns.

- To assign scores to columns of base pair rows, we go through different strategies based on the type of participating base pairs:

  - If base pair alphabets $\{<, >, [,], \{, \}\}$ are aligned with each other, we use scores from the corresponding RIBOSUM matrix.

  - If base pair alphabets {E,F,I,J,L,O} are aligned with each other, we use a quarter (1/4) of corresponding base pair scores from RIBOSUM matrix (Based on our observations, base pair alphabet Z is never seen in any grammar strings).

  - If a member of $\{<, >, [,], \{, \}\}$ is aligned with a member of $\{E, F, I, J, L, O\}$, we use half of the corresponding base pair score from RIBOSUM matrix.

  - The remaining columns of a base pair row are filled in with very large negative numbers.

- The alignment score of two bifurcations (#) or two end of derivations (|) is set to a very large positive number. The remaining columns of such rows are filled in with very large negative numbers.

We refer to Appendix B for both old and new score tables. We also need to set the gap penalties to perform pairwise alignment. Our gap penalty consists of open and extension gap penalties. The open penalty is assigned to the first matched gap with a grammar string alphabet. If after this gap we have to match another gap with a grammar alphabet then the negative gap extension score is used. We set gap penalties and extension empirically. In Table 4.1, each data set is presented along with the corresponding score table and gap penalties used in pairwise alignment.

The implementation of pairwise alignment is done by our colleague Achawanantakun and we refer the interested reader to [37] for more details on the subject.

Table 4.1: Mapping in the consensus sequence

Data sets	Score tables	Gap Penalties
Old conserved	clans Old score table	Open -8 / Ext -2
New conserved	clans New score table	Open -8 / Ext -2
Old Rfam-Clans	Old score table	Open -1 / Ext -12
New Rfam-Clans	New score table	Open -1 / Ext -15

4.2.4 Applying MCL clustering approach

The MCL (Markov Cluster Algorithm) is an unsupervised clustering algorithm for networks. This algorithm which is claimed to be fast and scalable originates from simulation of stochastic flow in graphs. Stijn van Dongen invented this algorithm which applies to different areas, especially in bioinformatics. The algorithm simulates flow using algebraic operations on matrices, namely expansion and inflation. The former operation coincides with normal matrix multiplication to model the spreading out of flow and its homogeneity. The latter (inflation) models the contraction of flow and its thickness/thinness in regions of higher/lower current. The MCL process causes flow to spread out within natural clusters and evaporate in between

different clusters. See [43] for mathematical theory behind the MCL algorithm. MCL has $O(N * k^2)$ time complexity in worst case, where N is the number of nodes in the graph, and k is the maximum number of neighbors tracked during computations.

We use pairwise alignment scores as inputs to MCL clustering application. Our assumption is that the pairwise alignment score between each two grammar strings shows the degree of similarity between their grammar strings and consequently between their secondary structures. Before feeding pairwise alignment scores of a data set to MCL, we first find the minimum score of data set and map it to the origin (zero). All other scores of the corresponding set are accordingly shifted.

We run MCL application on a set of pairwise alignment scores with various cut-off thresholds. The cut-off threshold is used to eliminate scores less than this threshold. This option helps us limit our search domain especially when we deal with large scale data sets.

4.3 Experimental results

4.3.1 Experimental results of old designed grammar string

Data set: conserved clans

In Figure 4.2, conserved clans along with their Rfam family members are shown.

We calculate the average of pairwise alignment shifted scores between Rfam family members within each clan. This value is called "internal average score". Table 4.2 shows classified conserved clans based on the internal average score factor. Experimental results show that clustering process is more successful in identifying clans whose members have higher internal average score than 4. We first analyze the behavior of Rfam families which belong to these clans during clustering process. To begin with, we name and also explain some figures and tables that will heavily be used in our analysis.

- Figure 4.3 illustrates the summary of a clustering process (presented in five levels)

Clan name	Clan family members	Description
CL00015	RF01376 RF01377 RF01318	CRISPR
CL00010	RF00008 RF00163	ribozyme
CL00088	RF00685 RF00794	snoRNA; CD-box;
CL00099	RF00643 RF00692	snoRNA; CD-box;
CL00073	RF00046 RF00609	snoRNA; CD-box;
CL00074	RF00186 RF00339	snoRNA; CD-box;
CL00076	RF00610 RF01280	snRNA;
CL00044	RF00581 RF01249	snoRNA; CD-box;
CL00048	RF00569 RF01183	snoRNA; CD-box;
CL00050	RF00087 RF00136	snoRNA; CD-box;
CL00064	RF00151 RF00608	snRNA;
CL00067	RF00270 RF01170 RF01200	snRNA;
CL00072	RF00055 RF01299	snRNA;
CL00008	RF00206 RF01277	snoRNA; CD-box;

Figure 4.2: Pairwise alignment configurations

Table 4.2: Classification of conserved clans based on internal average score

Average score interval	[1,2)	[2,3)	[3,4)	[4,5)	[5,6)
Conserved Clan families	CL00099	CL00008 CL00044 CL00088	CL00067 CL00064 CL00048 CL00073	CL000072 CL00050 CL00076 CL00076 CL00074 CL00015	CL00010

applied to Rfam family members of conserved clans. These clans all have internal average score of more than 4, i.e, two most right columns in Table 4.2, except for Rfam families shown in rectangles (in levels 3 and 4) which do not belong to the two most

right columns. We will later explain why this is the case.

- The table shown in Figure 4.5 presents shifted pairwise alignment scores between each two participating Rfam families in the clustering process. We fill in diagonal elements of the table with zeros as pairwise alignment of each family with itself is not defined. The two most right columns are also filled in with zeros. This will be taken up later in this section.

- Table 4.3 presents grammar patterns of conserved clans with internal average score higher than 4.

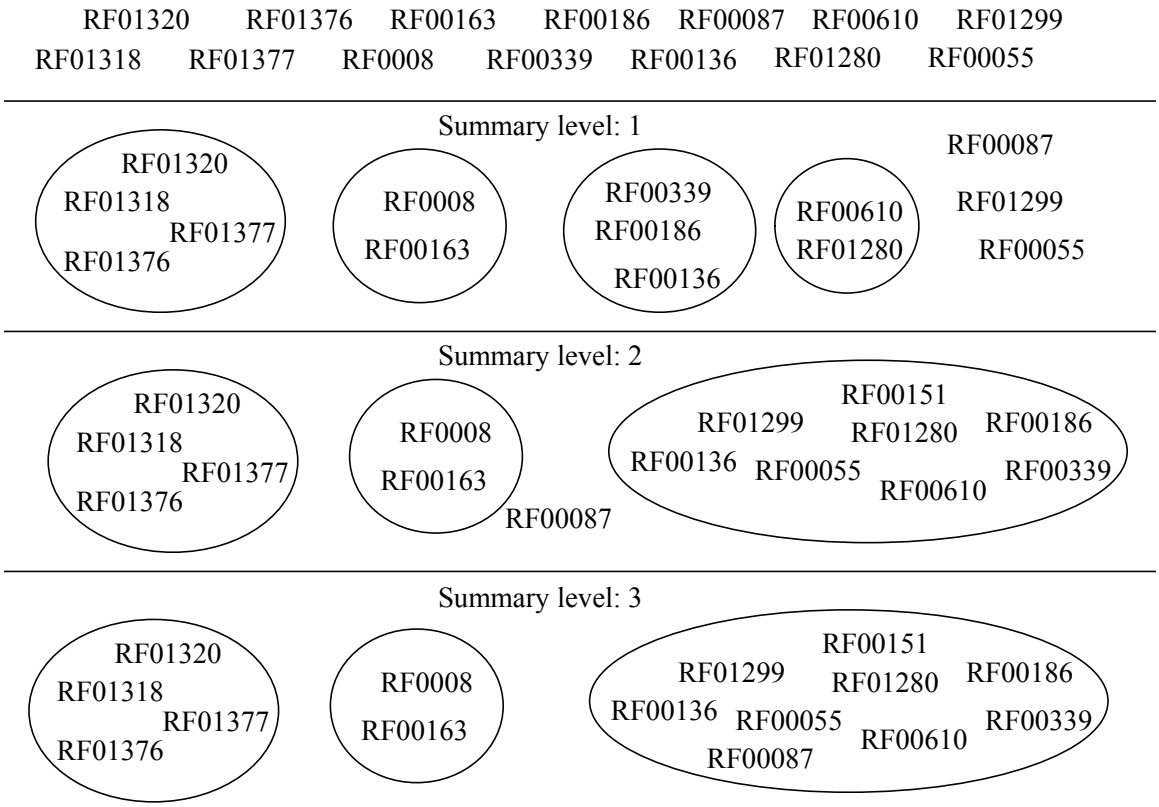


Figure 4.3: Summary of clustering process of conserved clans members

		CL00015				CL00074		CL00076	
		RF01320	RF01376	RF01377	RF01318	RF00186	RF00339	RF00610	RF01280
CL00015	RF01320	0	4.07	5.38	4.74	1.47	1.4	1.39	1.24
	RF01376		0	3.11	5.64	1.42	1.42	1.24	1.17
	RF01377			0	3.39	1.92	1.87	1.82	1.64
	RF01318				0	1.15	1.22	1.16	0.98
CL00074	RF00186					0	4.99	4.38	3.8
	RF00339						0	4.38	3.84
CL00076	RF00610							0	4.67
	RF01280								0
CL00050	RF00087								
	RF00136								
CL00072	RF00055								
	RF01299								
CL00010	RF00008								
	RF00163								

Figure 4.4: Pairwise alignment scores - Part1

		CL00050		CL00072		CL00010	
		RF00087	RF00136	RF00055	RF01299	RF00008	RF00163
CL00015	RF01320	1.545381	1.497188	1.256737	1.520346	0	0
	RF01376	1.25098	1.35261	1.24301	1.403463	0	0
	RF01377	1.605622	1.891358	1.500366	1.8	0	0
	RF01318	1.074509	1.164285	0.963441	1.15671	0	0
CL00074	RF00186	3.968627	4.461044	3.820689	3.874796	0	0
	RF00339	3.169697	4.086274	3.989855	3.916049	0	0
CL00076	RF00610	3.567816	3.765891	4.177011	3.634599	0	0
	RF01280	3.511355	3.78608	4.064583	3.302564	0	0
CL00050	RF00087	0	4.122222	3.252083	3.226515	0	0
	RF00136		0	3.764912	4.092857	0	0
CL00072	RF00055			0	4.349425	0	0
	RF01299				0	0	0
CL00010	RF00008					0	5.690196
	RF00163						0

Figure 4.5: Pairwise alignment scores - Part2

Table 4.3: Grammar patterns of 6 Clans

	Rfam families	Strategy 1	Strategy 2	Strategy 3 (P: 0.5, A:0.7, a:0.7)
CL00015	RF01320 RF01376 RF01377 RF01318	P| P| P| P|	P| P| P| P|	AaPA| AP| AaPA| AaP|
CL00010	RF00008 RF00163	P#P|P| P#P|P|	P#P|P| P#P|P|	Pa#aPA|PA| PA#aPA|P|
CL00074	RF00186 RF00339	P| P|	P| P|	PA| PA|
CL00076	RF00610 RF01280	P| P|	P| P|	PA| PA|
CL00050	RF00087 RF00136	P| P|	P| P|	PA| PA|
CL00072	RF00055 RF01299	P| P|	P| P|	PA| PA|

A. CL00010 and CL00015 clans

CL00010 and CL0015 are both identified and remained very well-conserved during clustering process (Figure 4.3). We study these two clans and investigate why our method could also put them together in the clustering process, very much like what clan database did.

CL00010 is called "Hammerhead clan" in Clan database. This clan carries two RNA families Hammerhead_1(RF0008) and Hammerhead_3(RF00163). Homology between these families has been established in the published literature [7,26]. In clustering process CL00010 family members are clustered together and remained isolated from other Rfam families. The reason is that although RF0008 and RF00163 share a similar secondary structure with each other, they are very different from other participants in structural level. RF0008 and RF00163 are the only two Rfam families in the Table 4.3 which have bifurcation (#) in their grammar patterns and as a result their pairwise alignments with other families result in large negative scores. Based on our own defined rule in pairwise alignment application,

these alignment types are filtered out and do not appear at the output. As a result, these families do not participate in clustering process and remain isolated from other families.

Clan CL00015 is also called "CRISPR-2 clan" and contains RNA families CRISPR-DR5 (RF01318), CRISPR-DR7 (RF01320), CRISPR-DR63 (RF01376) and CRISPR-DR64 (RF01377). To verify the relationship between Rfam families of this clan we refer to [32, 34, 39]. CL00015 members are also identified and remained isolated from other Rfam families during the clustering process. However, CL00015 clan members are not far from other participants in structural level that much. As it is seen in the grammar pattern Table 4.3, members in this clan share the same grammar patterns as other participants (excluding RF0008 and RF00163). Members of CL00015 have long chains of individual bases starting at both ends 3' and 5'. CL00074, CL00076, CL00050, and CL00072 clan members do not have this attribute. As a result, this clan family members could share a high score pairwise alignment with each other, have low pairwise alignment with other Rfam families, and finally are isolated from other clusters. For more details, one can see the table in Figure 4.5, and check the pairwise alignment scores of CL00015 family members with inside and outside of clan Rfam families.

B. CL00074 and CL00076 clans

CL00074 and CL00076 are both identified in clustering process (Figure 4.3), but are not remained isolated. We start with a short biological introduction of these clans and next will analyze the outcome of the experimental results for them.

CL00074 is called "SNORD101 clan" and contains two RNA families SNORD101 (RF000186) and snoR60 (RF00339). Small nucleolar RNAs snoR60 (RF00339) and snoRD101 (RF000186) both are non-coding RNA (ncRNA) molecules which function to modify other small nuclear RNAs (snRNAs). They both belong to the C/D box class of snoRNAs containing the conserved sequence motifs known as the C box (UGAUGA) and the D box (CUGA). Bioinformatic methods discussed in [32, 34, 39] confirm the relationship between these families.

Clan CL00076 is called "SNORD110" and contains two RNA families SNORD110 (RF00610) and snoR14 (RF01280). Functionality of both families is to modify other small nuclear RNAs (snRNAs). They also belong to C/D box class of snoRNAs which contain the C (UGAUGA) and D (CUGA) box motifs. Bioinformatic methods discussed in [24] confirm the relationship between these families.

Based on the Table 4.3, these two clans not only share very similar secondary structures with each other, but also they are similar to CL00050 and CL00072 clans in this aspect. Instead of high degree of similarity between this clan members with other participating Rfam families in the structural level, our clustering approach is still successful to identify CL00074 and CL00076 clans. Also in the pairwise alignment shown in Figure 4.5, it is clearly seen that members of CL00074 or CL00076 share higher pairwise alignment score with their clan-mates than those of other clans. Furthermore, although secondary structures of these clans are very similar to each other and to the members of other clans, designed grammar strings and score table are still successful to distinguish between Rfam family members of these two clans. However, in further steps of the clustering, RF00136 is merged with CL00074 clan members. This observation will be discussed in the next step.

C. CL00050 clan

CL00050 is also called "SNORD26 clan" and contains two RNA families SNORD26 (RF00087) and SNORD81 (RF00136). Small nucleolar RNAs SNORD26 and SNORD81 both are members of the C/D class of snoRNA and contain the C (UGAUGA) and D (CUGA) box motifs. Bioinformatic methods discussed in [32, 39] confirm the relationship between these families.

During the clustering process, RF00136 joins the cluster of CL00074 Rfam family members. The reason is that RF00136 is more similar to RF00186 member of CL00074 in both sequence and structure level. To see this fact better, please visit the corresponding pages in Clan database which show the secondary structure of CL00074 and CL00050 (Appendix C). In the following, pairwise alignments of {RF00136,RF00186} and {RF00136,RF00087} are shown, respectively. As it is seen, RF00136 has a better alignment with RF00186 than its clan-mate RF00087.

```
PPPPP---AAUGAUGA-CUUW----AAUUGUCGGAUACCCCUUCACUCCYU--YYA--UGAGUGA
PPPPPPPPCA-UGAUGAUCUCACYCCAACU-U--GA-ACUCUCUCAC----UGAUUACUUGA-UGA
-AACAUAAYAG-UCUGA|
YAAUA-AA-AGAUCUGA|
score= 243.00
align length= 83
RF00186-RF00136 pairwise alignment score before shifting:2.927711
RF00186-RF00136 pairwise alignment score after shifting:4.461044

cPPPP------GGGGAUGAUUUYA----AGAAC-UGAACUCUCU--CU--UU-C-UGAUGGWUUA
-PPPPPPPCAUG---AUGAUCUCACYCCA--ACUUGAACUCUCUCACUGAUUACUUGAUGA--YA
GUGGAGAAAASMMAAAAWMUCUGA|
AU----AAAA---GA----UCUGA|
score= 233.00
align length= 90
RF00087-RF00136 pairwise alignment score before shifting:2.588889
```

```
RF00087-RF00136 pairwise alignment score after shifting:4.122222
```

D. CL00072 clan

CL00072 is also called "SNORD96 clan" in Clan database and contains the RNA families SNORD96 (RF00055) and snR39B (RF00608). They are both members of the C/D class of snoRNA and contain the C (UGAUGA) and D (CUGA) box motifs [29].

Rfam family members of CL00072 are clustered together along with RF00151 Rfam family which belongs to CL00064. CL00064 is called "SNORD58 clan" and contains RNA families SNORD58 (RF00151) and SNORD99 (RF00608). These families are both non-coding RNA (ncRNA) molecules whose functionality is to modify other small nuclear RNAs (snRNAs). These both belong to the C/D box class of snoRNAs and contain the C (UGAUGA) and D (CUGA) box motifs. Bioinformatic methods discussed in [32, 39] confirm the relationship between these families.

RF00151 joins the cluster of CL00072 clan members because this Rfam family is more similar to RF01299 Rfam family in sequential level than RF00608 which is its clan-mate. To see this fact better, please visit the corresponding pages in Clan database which show the secondary structure of CL00072 and CL00064 (Appendix C). In the following, pairwise alignments of {RF00608, RF00151,RF01299} are shown, respectively. As it is seen, RF00151 has a better pairwise alignment with RF01299 than its clan-mate RF00608.

```
cPPPP----WGUGAUGA---CU-WUCUUA--GGACACCUUUGGAU--UA---AC---CAUGAAAAS
yPPPPPUCMAG-GAUGAAACCYAAUYUSAGUGGACAUCUMUGGAUGAWAMWUGCGGAUAUGGGA--
AAAWMWRUUCU--GA|
---------CUGAGA|
score= 172.00
align length= 82
RF00151-RF00608 pairwise alignment score before shifting:2.588889
RF00151-RF00608 pairwise alignment score after shifting:3.630894
```

```
cP--PPPWGUGAUGACUWUCUUAGGA-CACCUUU--GGAUUAACC----AUGAAAASAA-AWMWRUU
-PPPPPPWAUGAUGGC------A-WAYCAUCUUUCGGGACUGACCUGAAAUGRAGAGAWUAYWYAUU
-CUGA|
GCUGA|
score= 208.00
align length= 73
RF00151-RF01299 pairwise alignment score before shifting:2.849315
RF00087-RF00136 pairwise alignment score after shifting:4.382648
```

E. Conserved clans with internal average score less than 4

The clustering process is not successful to identify clans whose members have internal average score (Section 4.3.1) less than 4. We calculate the average of pairwise alignment scores between members of each clan and call it "external average score". In Figure 4.6, both internal and external average scores of 14 participating clans in this experiment are presented. Clans whose internal and external average scores are much closer to each other, have a lesser chance of being identified or remained isolated during clustering process. Clans whose members could not stick to their clan-mates and create their own isolated clans have close external and internal average scores. Also, their internal and external scores are close to the corresponding average scores of other clans.

Clustering results of Rfam−Clan data set

By applying the clustering approach to the shifted pairwise alignment scores of Rfam−Clan grammar strings, we could find some clusters which remain well-conserved. These clusters are presented in Table 4.4 following with a short description of the common biological functionality for members of each cluster.

- RF01112 and RF01083 are needed for efficient transcription. For more details on biological functionalities of RF01112 and RF01083, please visit the corresponding pages in Clan database (Appendix C).

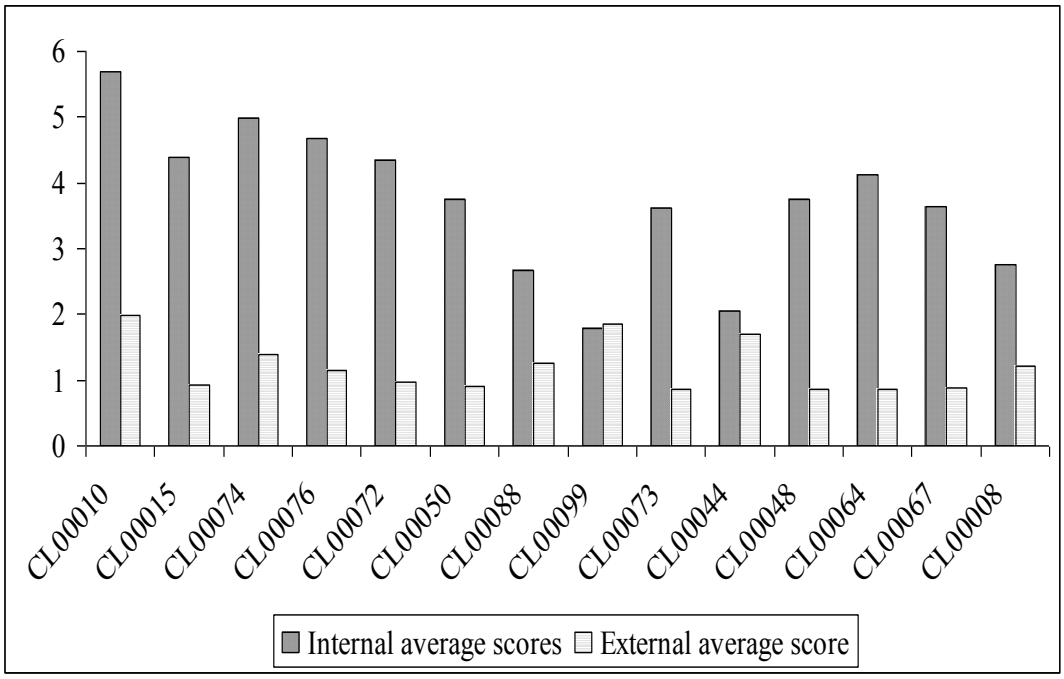


Figure 4.6: External and internal average scores of all conserved clans

- RF01111 and RF01109 are needed for efficient transcription. For more details on biological functionalities of RF01112 and RF01083, please visit the corresponding pages in Clan database (Appendix C).

- RF01025 and RF01017 have multiple roles in negative regulation (transcript degradation and sequestering, translational suppression) and possible involvement in positive regulation (transcriptional and translational activation). For more details on biological functionalities of RF01112 and RF01083, please visit the corresponding pages in Clan database (Appendix C).

- RF01386 and RF00014 are used in translation and transcription terminator. For more details on biological functionalities of RF01112 and RF01083, please visit the corresponding pages in Clan database (Appendix C).

Table 4.4: Discovered Clusters in Rfam−Clan

Number	Rfam families	Grammar pattern	RNA Type	Domain
1	RF01112 RF01083	APA| APA|	Cis-reg	Viruses
2	RF01111 RF01109	aA| aA|	Cis-reg	Viruses Viruses, Bacteria
3	RF01025 RF01017	PA| PA|	miRNA	Eukaryota
4	RF01386 RF00014	APA| APA|	sRNA	Bacteria Bacteria, Eukaryota
5	RF01340 RF01322	aP| aP|	CRISPR	Bacteria
6	RF01115 RF01106	APA| APA|	Cis-reg	Viruses Viruses, Eukaryota

- RF01340 and RF01322 function as a prokaryotic immune system. For more details on biological functionalities of RF01112 and RF01083, please visit the corresponding pages in Clan database (Appendix C).

- RF01115 and RF01106 are needed for efficient transcription. For more details on biological functionalities of RF01112 and RF01083, please visit the corresponding pages in Clan database (Appendix C).

4.3.2 Analysis of experimental results based on using new version of grammar string design

By applying clustering process to the shifted pairwise alignment of conserved clans which are generated based on new grammar string design, the results are the same as we achieved before in Section 4.3.1. We can conclude that the performance of old designed grammar string is as well as the new one. Designing a grammar string with distinguished base pairs

or defining a new score table could not enhance our ability to do a better clustering. We have done the same experiment for the new grammar string design, however, the questions as to how to interpret the results require more investigation and are parts of the future work on this subject.

APPENDICES

Appendix A

Ribosum matrix

A.1 Used in grammar string chapter

A	2.22			
C	-1.86	1.16		
G	-1.46	-2.48	1.03	
U	-1.39	-1.05	-1.74	1.65
	A	C	G	U

Figure A.1: Ribosum matrix - Part1

AA	-2.49						
AC	-7.04	-2.11					
AG	-8.24	-8.89	-0.8				
AU	-4.32	-2.04	-5.13	4.49			
CA	-8.84	-9.37	-10.41	-5.56	-5.13		
CC	-14.37	-9.08	-14.53	-6.71	-10.53	-3.59	
CG	-4.68	-5.86	-4.57	1.67	-3.57	-5.71	5.36
CU	-12.64	-10.45	-10.14	-5.17	-8.49	-5.77	-4.96
GA	-6.86	-9.73	-8.61	-5.33	-7.98	-12.43	-6
GC	-5.03	-3.81	-5.77	2.7	-5.95	-3.7	2.11
GG	-8.39	-11.05	-5.38	-5.61	-11.36	-12.58	-4.66
GU	-5.84	-4.72	-6.6	0.59	-7.93	-7.88	-0.27
UA	-4.01	-5.33	-5.43	1.61	-2.42	-6.88	2.75
UC	-11.32	-8.67	-8.87	-4.81	-7.08	-7.4	-4.91
UG	-6.16	-6.93	-5.94	-0.51	-5.63	-8.41	1.32
UU	-9.05	-7.83	-11.07	-2.98	-8.39	-5.41	-3.67
	AA	AC	AG	AU	CA	CC	CG

Figure A.2: Ribosum matrix - Part2

CU	-2.28								
GA	-7.71	-1.05							
GC	-5.84	-4.88	5.02						
GG	-13.69	-8.67	-4.13	-1.98					
GU	-5.61	-6.1	1.21	-5.77	3.47				
UA	-4.72	-5.85	1.6	-5.75	-0.57	4.97			
UC	-3.83	-6.63	-4.49	-12.01	-5.3	-2.98	-3.21		
UG	-7.36	-7.55	-0.08	-4.27	-2.09	1.14	-4.76	3.36	
UU	-5.21	-11.54	-3.9	-10.79	-4.45	-3.39	-5.97	-4.28	-0.02
	CU	GA	GC	GG	GU	UA	UC	UG	UU

Figure A.3: Ribosum matrix - Part3

Appendix B

Old and new score tables

B.1 Old grammar string score table

Our website: http://www.cse.msu.edu/ yannisun/grammar-string

	A	U	C	G
A	7	-12	-12	-1
U	-12	7	-1	-12
C	-12	-1	7	-12
G	-1	-12	-12	7
a	-1000000	-1000000	-1000000	-1000000
u	-1000000	-1000000	-1000000	-1000000
c	-1000000	-1000000	-1000000	-1000000
g	-1000000	-1000000	-1000000	-1000000
P	-100	-100	-100	-100
#	-1000000	-1000000	-1000000	-1000000
|	-1000000	-1000000	-1000000	-1000000

Figure B.1: Score table used in pairwise alignments of old garment strings - Part1

	a	u	c	g	P	#	|
A	-1000000	-1000000	-1000000	-1000000	-100	-1000000	-1000000
U	-1000000	-1000000	-1000000	-1000000	-100	-1000000	-1000000
C	-1000000	-1000000	-1000000	-1000000	-100	-1000000	-1000000
G	-1000000	-1000000	-1000000	-1000000	-100	-1000000	-1000000
a	7	-12	-12	-1	-100	-1000000	-1000000
u	-12	7	-1	-12	-100	-1000000	-1000000
c	-12	-1	7	-12	-100	-1000000	-1000000
g	-1	-12	-12	7	-100	-1000000	-1000000
P	-100	-100	-100	-100	7	-1000000	-1000000
#	-1000000	-1000000	-1000000	-1000000	-1000000	72	-1000000
|	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000	58

Figure B.2: Score table used in pairwise alignments of old garment strings - Part2

B.2 New grammar string score table

	A	U	C	G	a	u	c
A	2.22	-1.39	-1.86	-1.46	-1000000	-1000000	-1000000
U	-1.39	1.65	-1.05	-1.74	-1000000	-1000000	-1000000
C	-1.86	-1.05	1.16	-2.48	-1000000	-1000000	-1000000
G	-1.46	-1.74	-1.48	1.03	-1000000	-1000000	-1000000
a	-1000000	-1000000	-1000000	-1000000	2.22	-1.39	-1.86
u	-1000000	-1000000	-1000000	-1000000	-1.39	1.65	-1.05
c	-1000000	-1000000	-1000000	-1000000	-1.86	-1.05	1.16
g	-1000000	-1000000	-1000000	-1000000	-1.46	-1.74	-1.48
<	1.11	-2.75	-3.72	-2.92	-2.78	0.825	-2.1
>	-2.78	0.825	-2.1	-3.48	1.11	-2.75	-3.72
[	-3.72	-2.1	-0.58	-4.96	-2.92	-3.48	-4.96
]	-2.92	-3.48	-4.96	0.515	-3.72	-2.1	-0.58
(	-2.92	-3.48	-4.96	0.515	-2.78	0.825	-2.1
)	-2.78	0.825	-2.1	-3.48	-2.92	-3.48	-4.96
E	0.555	-5.56	-7.44	-5.84	-5.56	0.412	-4.2
F	-5.56	0.412	-4.2	0.29	0.555	-5.56	-7.44
I	-7.44	-4.2	0.29	-9.92	-5.84	-6.69	-9.92
J	-5.84	-6.69	-9.92	0.257	-7.44	-4.2	0.29
L	-5.84	-6.69	-9.92	0.257	-5.56	0.412	-4.2
O	-5.56	0.412	-4.2	-6.96	-5.84	-6.69	-9.92
Z	-13	-13	-13	-13	-13	-13	-13
#	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000
|	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000

Figure B.3: Score table used in pairwise alignments of old garment strings - Part1

	g	<	>	[	]	(	)
A	-1000000	1.11	-2.75	-3.72	-2.92	-2.92	-2.78
U	-1000000	-2.78	0.825	-2.1	-3.48	-3.48	0.825
C	-1000000	-3.72	-2.1	-0.58	-4.96	-4.96	-2.1
G	-1000000	-2.92	-3.48	-4.96	0.515	0.515	-3.48
a	-1.46	-2.78	1.11	-2.92	-3.72	-2.78	-2.92
u	-1.74	0.825	-2.78	1.11	-2.1	0.825	-3.48
c	-2.48	-2.1	-3.72	-4.96	0.58	-2.1	-4.96
g	1.03	-3.48	-2.92	0.515	-4.96	-3.48	0.515
<	-3.48	4.49	1.61	1.67	2.7	0.59	-0.51
>	-2.92	1.61	4.97	2.75	1.6	-0.57	1.14
[	0.515	1.67	2.75	5.36	2.11	-0.27	1.32
]	-4.96	2.7	1.6	2.11	5.62	1.21	-0.08
(	-3.48	0.59	-0.57	-0.27	1.21	3.47	-2.09
)	0.515	-0.51	1.14	1.32	-0.08	-2.09	3.36
E	0.29	2.25	0.805	0.835	1.35	0.295	-1.02
F	-5.84	0.805	2.485	1.375	0.8	-1.14	0.57
I	0.257	0.835	1.375	2.68	1.055	-0.54	0.66
J	-9.92	1.35	0.8	1.055	2.81	2.42	-0.16
L	-6.96	0.295	-1.14	-0.54	2.42	1.735	-4.18
O	0.257	-1.02	0.57	0.66	-0.16	-4.18	1.68
Z	-13	-10	-10	-10	-10	-10	-10
#	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000
|	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000

Figure B.4: Score table used in pairwise alignments of old garment strings - Part2

	E	F	I	J	L	O	Z
A	0.555	-5.56	-7.44	-5.84	-5.84	-5.56	-13
U	-5.56	0.412	-4.2	-6.96	-6.96	0.412	-13
C	-7.44	-4.2	0.29	-9.92	-9.92	-4.2	-13
G	-5.84	-6.96	-9.92	0.257	0.257	-6.96	-13
a	-5.56	0.555	-5.84	-7.44	-5.56	-5.84	-13
u	0.412	-5.56	-6.96	-4.2	0.412	-6.96	-13
c	-4.2	-7.44	-9.92	0.29	-4.2	-9.92	-13
g	-6.96	-5.84	0.257	-9.92	-6.96	0.257	-13
<	2.25	0.825	0.835	1.35	0.295	-1.02	-3
>	0.805	2.485	1.375	0.8	-1.14	0.57	-3
[	0.835	1.375	2.68	1.055	-0.54	0.66	-3
]	1.35	0.8	1.055	2.81	0.605	-1.6	-3
(	0.295	-1.14	-0.54	0.605	1.735	-4.18	-3
)	-1.02	0.57	0.66	-0.16	-4.18	1.68	-3
E	1.125	0.402	0.417	0.675	0.147	-2.04	-3
F	0.402	1.242	0.687	0.4	-2.28	0.285	-3
I	0.417	0.687	1.34	0.527	-1.08	0.33	-3
J	0.675	0.4	0.527	1.405	1.21	-0.32	-3
L	0.147	-2.28	-1.08	1.21	0.867	-8.36	-3
O	-2.04	0.285	0.33	-0.32	-8.36	0.84	-3
Z	-10	-10	-10	-10	-10	-10	-10
#	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000
|	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000	-1000000

Figure B.5: Score table used in pairwise alignments of old garment strings - Part3

	#	|
A	-1000000	-1000000
U	-1000000	-1000000
C	-1000000	-1000000
G	-1000000	-1000000
a	-1000000	-1000000
u	-1000000	-1000000
c	-1000000	-1000000
g	-1000000	-1000000
<	-1000000	-1000000
>	-1000000	-1000000
[	-1000000	-1000000
]	-1000000	-1000000
(	-1000000	-1000000
)	-1000000	-1000000
E	-1000000	-1000000
F	-1000000	-1000000
I	-1000000	-1000000
J	-1000000	-1000000
L	-1000000	-1000000
O	-1000000	-1000000
Z	-1000000	-1000000
#	100	-1000000
|	-1000000	80

Figure B.6: Score table generated based on Ribosum matrix and used in pairwise alignments of new garment strings - Part4

Appendix C

Links to Web Pages of Clan families and their members

C.1 Used in experimental results section

Name	Webpage
CL00074	http://rfam.sanger.ac.uk/clan/CL00074
CL00050	http://rfam.sanger.ac.uk/clan/CL00050
CL00072	http://rfam.sanger.ac.uk/clan/CL00072
CL00064	http://rfam.sanger.ac.uk/clan/CL00064
RF01112	http://rfam.sanger.ac.uk/family/RF01112
RF01083	http://rfam.sanger.ac.uk/family/RF01083
RF01111	http://rfam.sanger.ac.uk/family/RF01111
RF01109	http://rfam.sanger.ac.uk/family/RF01109
RF01025	http://rfam.sanger.ac.uk/family/RF01025
RF01017	http://rfam.sanger.ac.uk/family/RF01017
RF01386	http://rfam.sanger.ac.uk/family/RF01386
RF00014	http://rfam.sanger.ac.uk/family/RF00014
RF01340	http://rfam.sanger.ac.uk/family/RF01340
RF01322	http://rfam.sanger.ac.uk/family/RF01322
RF01115	http://rfam.sanger.ac.uk/family/RF01115
RF01106	http://rfam.sanger.ac.uk/family/RF01106

Figure C.1: Links to Web Pages

BIBLIOGRAPHY

Bibliography

[1] Rujira Achawanantakun, Seyedeh Shohreh Takyar, and Yanni Sun. Grammar string: a novel ncRNA secondary structure representation. In *9th Annual International Conference on Computational Systems Bioinformatics*, pages 2–13, Stanford, CA, August 2010.

[2] S F Altschul, W Gish, W Miller, E W Myers, and D J Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[3] F Bai, D Li, and T Wang. A new mapping rule for RNA secondary structures with its applications. *J. Math. Chem.*, 43:932–943, 2008.

[4] M. Blanchette, W. J. Kent, C. Riemer, L. Elnitski, A. F. Smit, K. M. Roskin, R. Baertsch, K. Rosenbloom, H. Clawson, E. D. Green, D. Haussler, and W. Miller. Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res*, 14(4):708–715, 2004.

[5] Francis Crick. Central dogma of molecular bilogy. *Nature*, 227:561–563, August 1970.

[6] Neil D. Rawlings and Alan J. Barrett. Evolutionary families of peptidases. *Biochem J.*, 290:205–218, 1993.

[7] Ruffner DE, Stormo GD, and Uhlenbeck OC.

[8] Robin Dowell and Sean Eddy. Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC Bioinformatics*, 5(1):71, 2004.

[9] Robin D Dowell and Sean R Eddy. Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics*, 7(400), 2006.

[10] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, UK, 1998.

[11] Robert D. Finn, Jaina Mistry, Benjamin Schuster-Bockler, Sam Griffiths-Jones, Volker Hollich, Timo Lassmann, Simon Moxon, Mhairi Marshall, Ajay Khanna, Richard Durbin, Sean R. Eddy, Erik L. L. Sonnhammer, and Alex Bateman. Pfam: clans, web tools and services. *Nucleic Acids R.*, 34:247–251, 2006.

[12] P. Gardner, J. Daub, J. Tate, B. Moore, I. Osuch, S. Griffiths-Jones, R. Finn, E. Nawrocki, D. Kolbe, S. Eddy, and A. Bateman. Rfam: Wikipedia, clans and the decimal release. *Nucleic Acids Res.*, 39(Database):D141–D145, 2010.

[13] S. Griffiths-Jones, S. Moxon, M. Marshall, A. Khanna, S. R. Eddy, and A. Bateman. Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res.*, 33:D121–D124, 2005.

[14] M. Höchsmann, B. Voss, and R. Giegerich. Pure multiple RNA secondary structure alignments: a progressive profile approach. *IEEE/ACM Trans Comput Biol Bioinform*, 1(1):53–62, 2004.

[15] Matthias Höchsmann, Thomas Töller, Robert Giegerich, and Stefan Kurtz. Local Similarity in RNA Secondary Structures. In *CSB '03: Proceedings of the IEEE Computer Society Conference on Bioinformatics*, page 159, Washington, DC, USA, 2003. IEEE Computer Society.

[16] I.L. Hofacker. Vienna RNA secondary structure server. *Nucleic Acids Res.*, 31:3429–3431, 2003.

[17] IL Hofacker, M Fekete, and PF Stadler.

[18] Ivo L. Hofacker, Stephan H. F. Bernhart, and Peter F. Stadler. Alignment of RNA base pairing probability matrices. *Bioinformatics*, 20(14):2222–2227, 2004.

[19] Ian Holmes. Accelerated probabilistic inference of RNA structure evolution. *BMC Bioinformatics*, 6(1):73, 2005.

[20] Tommi Jaakkola and David Haussler. Exploiting Generative Models in Discriminative Classifiers. In *NIPS*, pages 487–493, 1998.

[21] Ellis JC. and Brown JW. The RNase P family. *RNA BIOLOGY*.

[22] Urban JH. and Vogel J. Two seemingly homologous noncoding RNAs act hierarchically to activate glmS mRNA translation. *PLoS Biol.*, 6:e64, 2008.

[23] Matthew W. Jones-Rhoades, David P. Bartel, and Bonnie Bartel. MicroRNAs and Their Regulatory Roles in Plants. *Annual Review of Plant Biology*, 57:19–53, 2006.

[24] Brown JW, Echeverria M, Qu LH, Lowe TM, Bachellerie JP, Httenhofer A, Kastenmayer JP, Green PJ, Shaw P, and Marshall DF. Plant snoRNA database. *Nucleic Acids Res.*, 31:432–435, 2003.

[25] S Karlin and S F Altschul.

[26] Hertel KJ, Pardi A, Uhlenbeck OC, Koizumi M, Ohtsuka E, Uesugi S, Cedergren R, Eckstein F, Gerlach WL, and Hodgson R. Numbering system for the hammerhead. *Nucleic Acids Res.*, 20:32–52, 92.

[27] Robert J Klein and Sean R Eddy. *RSEARCH: Finding homologs of single structured RNA sequences*. BMC Bioinforma, USA, 2003.

[28] B. Knudsen and J. Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Res*, 31(13):3423–3428, 2003.

[29] Laurent Lestrade and Michel J. Weber. snoRNA-LBME-db, a comprehensive database of human H/ACA and C/D box snoRNAs. *Nucleic Acids Res.*, 34:158–162, 2006.

[30] C Li, A H Wang, and L Xing. Similarity of RNA secondary structures. *J. Comput. Chem.*, 28:508–512, 2007.

[31] Shanfa Lu, Rui Shi, Cheng-Chung Tsao, Xiaoping Yi, Laigeng Li, and Vincent L. Chiang. RNA silencing in plants by the expression of siRNA duplexes. *Nucl. Acids Res.*, 32(21):e171–, 2004.

[32] Martin Madera. Profile Comparer: a program for scoring and aligning profile hidden Markov models. *Bioinformatics*, 24(22):26302631, 2008.

[33] J. S. Mccaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29(6-7):1105–1119, 1990.

[34] Eric P. Nawrocki, Diana L. Kolbe, and Sean R. Eddy. Infernal 1.0: inference of RNA alignments. *Bioinformatics*, 25:13351337, 2009.

[35] Elena Rivas and Sean R. Eddy. Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics*, 2(1):8, 2001.

[36] David Sankoff. Simultaneous Solution of the RNA Folding, Alignment and Protosequence Problems. *SIAM Journal on Applied Mathematics*, 45(5):810–825, 1985.

[37] Needleman SB and Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48:443–453, 1970.

[38] Sven Siebert and Rolf Backofen. MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics*, 21(16):3352–3359, 2005.

[39] Eddy SR. A probabilistic model of local sequence alignment that simplifies statistical significance estimation. *PLoS Comput Biol.*

[40] Julie D. Thompson, Desmond G. Higgins, and Toby J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.*, 22(22):4673–4680, 1994.

[41] Elfar Torarinsson, Jakob H. Havgaard, and Jan Gorodkin. Multiple structural alignment and clustering of RNA sequences. *Bioinformatics*, 23(8):926–932, 2007.

[42] Helene Touzet and Olivier Perriquet. CARNAC: folding families of related RNAs. *Nucl. Acids Res.*, 32(suppl. 2):W142–145, 2004.

[43] Stijn van Dongen. *Graph clustering by flow simulation.* PhD thesis, University of Utrecht, May 2000.

[44] S. Washietl, I. L. Hofacker, and P. F. Stadler. Fast and reliable prediction of noncoding RNAs. *Proc Natl Acad Sci U S A*, 102(7):2454–2459, 2005.

[45] Sebastian Will, Kristin Reiche, Ivo L Hofacker, Peter F Stadler, and Rolf Backofen. Inferring Noncoding RNA Families and Classes by Means of Genome-Scale Structure-Based Clustering. *PLoS Comput Biol*, 3(4):e65, 2007.

[46] A. Wilm, I Mainz, and G. Steger. An enhanced RNA alignment benchmark for sequence alignment programs. *Algorithms Mol Biol.*, 1(19), 2006.

[47] Y Zhang, J Qiu, and Su L. Comparing RNA secondary structures based on 2D graphical representation. *Chemical Physics Letters*, 458:180–185, 2008.