# Stochastic Context-Free Grammars for Modeling Three Spliceosomal Small Nuclear Ribonucleic Acids

Rebecca Christine Underwood

UCSC-CRL-94-23
June 9, 1994

Baskin Center for
Computer Engineering & Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

ABSTRACT

In this thesis, stochastic context-free grammars (SCFGs) are applied to the problems of folding, aligning and modeling families of homologous RNA sequences—specifically, the small nuclear RNA sequences of the spliceosome. SCFGs generalize the hidden Markov models (HMMs) used in related work on protein and DNA to capture the sequences' common primary and secondary structure. This thesis discusses a recently introduced algorithm, Tree-Grammar EM, for deducing SCFG parameters automatically from unaligned, unfolded training sequences [SBU$^+$93, SBM$^+$94, SBH$^+$94a] and demonstrates its application to modeling three of the five prominent trans-acting factors in spliceosomal small nuclear RNA: `U1`, `U5` and `U6`. Tree-Grammar EM, a generalization of the HMM forward-backward algorithm, is based on tree grammars and is faster than the previously proposed inside-outside SCFG training algorithm. Results show that after having been trained on as few as about 20 snRNA sequences, each of the three models can discern snRNA sequences from similar-length RNA sequences of other kinds, can find secondary structure of new snRNA sequences and can produce multiple alignments of snRNA sequences.

**Keywords:** Stochastic context-free grammar, small nuclear RNA, multiple alignment, database search, folding sequences

# Contents

# List of Figures

## Acknowledgments

Foremost, I thank my co-workers on the SCFG project at UC Santa Cruz: Yasubumi Sakakibara, Michael Brown, Richard Hughey, I. Saira Mian, Kimmen Sjölander and my advisor David Haussler. I am indebted in particular to biologist Saira Mian for providing the snRNA data sets and alignments she developed with Christine Guthrie *et al.* I thank Tracy Larrabee for generously allowing me to use her Decstation 5000/240 computer, without which the results of this thesis would not have been obtainable. I am grateful to Leslie Grate, author of the graphical grammar-describing program `scfgedit`, which greatly simplified the task of designing my initial grammars. For supplying his graphical RNA-displaying program `xrna`, I thank Bryn Weiser. Thanks to Brad Gulko and Michael Brown for helpful discussions about Dirichlet distributions. For their thorough, insightful and timely commenting on this thesis, I express my gratitude to my reading committee—David Haussler, David Paul Helmbold and Kevin Karplus—and again to Saira Mian.

Most of all, I appreciate the sundry and sometimes ineffable advice, encouragement, kindness and sustenance provided by my family, especially Grace, Bob and Deborah, and my friends, a partial list of whom must include Ian Barland, Jacob Lewis, Stefan Helmreich, Jeroen Ritsema, Anne Urban, Madhavan Thirumalai, Saskia Goes, Saira Mian and Rob Antonucci. And thanks to Richard Hughey, June Sison and Joel Darnauer for the YATFAMPs and other *Star Trek* frolics.

Parts of the Introduction, Methods and Discussion sections are based on articles previously published as UC Santa Cruz technical reports [SBU+93, SBH+93], in the Proceedings of the 1994 Hawaii International Conference on System Sciences [SBM+94] and in Proceedings of the Asilomar Conference on Combinatorial Pattern Matching [SBH+94a], and recently submitted to *Nucleic Acids Research* [SBH+94b].

# 1. Introduction

## 1.1 Problem statement and context

Predicting the structure of macromolecules by theoretical or experimental means is a challenging problem. Efforts to sequence the genomes of organisms [SCH$^+$82, DSC83, SAB$^+$77, SCF$^+$78, DS81, DS83, BBB$^+$84, DPBB92, PBDB93, Oga93, OvAC$^+$92, SDT$^+$92, MAHK91, Joi93, Ols93, OS93, vDF$^+$92, Min93] and organelles [HSW$^+$89, HHD$^+$93, CC93, OYO$^+$92, THSH92, CMDM90, CRR$^+$89, GPD$^+$89, Sut79] have heightened awareness of the essential role of computers in data acquisition, management and analysis. The increasing numbers of DNA, RNA and protein sequences yielded by these projects [Cou91] provoke a growing need for developing new approaches in computational biology such as hidden Markov models (HMMs) [LG87, Chu89, Rab89, HKMS93, KBM$^+$94, BCHM93, CS92] and other methods [HSS93]. In addition to the accelerated discovery of sequences related by a natural phylogeny, the generation of "artificial" phylogenies by experimental design for proteins (reviewed in Arnold's paper [Arn93]) and RNA (reviewed in an article by Burke and Berzal-Herranz [BBH93]) serves only to exacerbate the problem of growth in sequence data. Determining common or consensus patterns among a family of sequences, producing a multiple sequence alignment, discriminating members of the family from non-members and discovering new members of the family will continue to be some of the most important and fundamental tasks in mathematical analysis and comparison of macromolecular sequences [DA89, Doo90]. (A *multiple alignment* of RNA sequences is a list of the sequences with the letters representing nucleotides spaced such that nucleotides considered equivalent have their letters appearing in the same column, or position, in the list. To enhance the alignment of some sequences with respect to others, spaces may need to be inserted in particular sequences.)

This thesis discusses the application of *stochastic context-free grammars* (SCFGs) to the problems of statistical modeling, multiple alignment, discrimination and prediction of the secondary structure of small nuclear RNA in the spliceosome. The work described here extends previous successful efforts to model, align and discriminate tRNA sequences [SBU$^+$93, SBM$^+$94, SBH$^+$93, ED94, SBH$^+$94a]. This approach is analogous to previous work on modeling protein families and domains and DNA with HMMs [HKMS93, KBM$^+$94, KMH93].

In ribonucleic acids (RNA), the nucleotides adenine (`A`), cytosine (`C`), guanine (`G`) and uracil (`U`) interact in specific ways to form characteristic *secondary-structure motifs* such as helices, loops and bulges [Sae84, WPT89]. Further folding and hydrogen-bonding interactions between remote regions orient these secondary-structure elements with respect to each other to form the functional system. Higher-order interactions with other proteins or nucleic acids may also occur. In general, however, the folding of an RNA chain into a functional molecule is largely governed by the formation of intramolecular `A`-`U` and `G`-`C` Watson–Crick base pairs[1] as well as `G`-`U` and, more rarely, `G`-`A` base pairs. Such base pairs constitute the so-called *biological palindromes* in the genome.[2]

[1] These are termed *base pairs* perhaps for historical reasons, not because they are chemically basic; rather, nucleic acids are acidic because of the presence of a negatively charged phosphate backbone, but they interact with chemically basic molecules [Mia94].

[2] They are called palindromes although the pairing halves do not reflect the identical bases but rather

Comparative analyses of two or more protein or nucleic-acid sequences have been used widely in detection and evaluation of biological similarities and evolutionary relationships. Several methods for producing these multiple sequence alignments have been developed, most based on dynamic programming techniques (for example, see works by Waterman [Wat89]). However, when RNA sequences are to be aligned, both the primary *and* secondary structure need to be considered since generation of a multiple sequence alignment and analysis of folding are mutually dependent. Elucidation of common folding patterns among two or more sequences may indicate the pertinent regions to be aligned and vice versa [San85].

Currently, there are two principal methods for predicting secondary structure of RNA, or which nucleotides are base-paired. Phylogenetic analysis of homologous RNA molecules [FW75, WGGN83] ascertains structural features that are conserved during evolution. It is based on the premise that functionally equivalent RNA molecules are also structurally equivalent and relies on alignment and subsequent folding of many sequences into similar secondary structures (see review papers [JOP89, WGGN83]). Such comparative methods have been used to infer the structure of small nuclear RNAs [GP88], tRNA [Lev69, HAE$^+$65, MEK66, ZDF$^+$66, RSF$^+$66], 5S RNA [FW75], 16S ribosomal RNA (rRNA) [WMG$^+$80, SCZ$^+$80, ZGB81], 23S rRNA [NKW$^+$91, GZB81, BKM$^+$81], group I introns [MW90, MECS90], group II introns [MUO89], ribonuclease P RNA [BHJ$^+$91, TE93], 7S RNA (signal recognition particle RNA) [Zwi89], telomerase RNA [RB91], MRP RNA [SBDC93] and TAR RNA of human and simian immunodeficiency viruses [Ber92]. The original procedure of Noller and Woese [NW81] detects compensatory base changes in putative helical elements: contiguous antiparallel arrangement of A-U, G-C and G-U pairings.[3] Positions that co-vary are assumed to be base-paired. This procedure was subsequently formalized into an explicit computer algorithm [WAG84, Wat88] that stores *all* "interesting" patterns, a potential problem as the number of patterns increases, where "interesting" is loosely defined as potential helical regions where there is evidence of more than one compensating base change, implying that the bases may be paired. The algorithm of Sankoff [San85] for simultaneously aligning and folding sequences is generally impractical in terms of time and space for large numbers of long sequences. Given an alignment of homologous RNA sequences, heuristic methods have been proposed to predict a common secondary structure [HK93, CK91, CZJ91]. However, there remains no reliable or automatic way of inferring an optimal consensus secondary structure even if the related sequences are already aligned. (A *consensus structure* for a group of RNA sequences represents the secondary structure most common to all of those sequences.) Because considerable manual intervention is still required to identify potential helices that maintain base complementarity, automation and development of more rigorous comparative analysis protocols are under continual development [GPH$^+$92, Lap92, KB93, Wat89, WOW$^+$90].

The second technique for predicting RNA secondary structure employs thermodynamics to compare the free energy changes predicted for formation of possible secondary structure and relies on finding the structure with the lowest free energy [TUL71, TSF88, Gou]. Such

---

the complementary bases. For instance, in the biological palindrome GUAC the first two bases GU pair with the last two bases AC as follows: G with C, U with A.

[3] Given a set of homologous RNA molecules, this process involves making an initial multiple alignment, marking *transitions* (a purine [A or G] becoming a pyrimidine [C or U]) and *transversions* (C becoming U, or G becoming A) between sequences, fixing the positions of "interesting" patterns, then iterating, producing a new multiple alignment and so forth, until helices are deduced [Mia94].

energy minimization depends on thermodynamic parameters and computer algorithms to evaluate the optimal and suboptimal free-energy folding of an RNA species (see review papers [JTZ90, ZS84]). (These thermodynamic parameters are obtained from actual experiments using small model compounds, where free energy changes for the formation of loops or other structures are measured.) To obtain a common folding pattern for a set of related molecules, Zuker has suggested predicting a folding for each sequence separately using these algorithms and then searching for a common structure [Zuk89a]. This method's limitations stem partially from the uncertainty in the underlying energy model, and the technique may be overly sensitive to point mutations.[4] Some researchers are attempting to combine both phylogenetic and energetic approaches [LZ91].

Using methods different from those described above, several groups have enumerated schemes or programs to search for patterns in proteins or nucleic acid sequences [Sta90, LWS87, SA90, AWM+84, SM87, GMC90, CAKF86, PC93]. String pattern-matching programs based on the UNIX `grep` function, developed in unpublished work by S. R. Eddy [STG92] and others [MMCN93], search for secondary structure elements in a sequence database. If there is prior knowledge about sequence and structural aspects of an RNA family, this can be employed to create a *descriptor* (discriminating pattern) for the family which can then be used for database searching or generating an alignment for the family. This has been demonstrated clearly for tRNA [FB91, Sta80, Mar86], where approximate string matching (locating all occurrences of substrings that are within a given similarity neighborhood of an exact match to the pattern) proved to be important.

The method of multiple alignment and folding used here differs markedly from the conventional techniques because it builds a statistical model *during* rather than *after* the process of alignment and folding. This approach has been applied to modeling tRNA sequences [SBH+93], and a similar approach has been applied to modeling protein families [HKMS93, KBM+94] and DNA [KMH93] with HMMs.

Though in principle HMMs could be used to model RNA, the standard HMM approach treats all positions in an alignment as having independent distributions and is unable to model the interactions between positions. However, if two positions in an alignment are base-paired, then the bases at these positions will be highly correlated. Since such base-pairing interactions play a dominant role in determining RNA structure and function, any statistical method for modeling RNA that does not consider these interactions will encounter insurmountable problems.

This thesis uses formal language theory to describe a means to generalize HMMs to model most of the interactions seen in RNA. As in the elegant work of Searls [Sea92], strings of characters representing pieces of DNA, RNA and protein are viewed as sentences derived from a formal grammar. In the simplest kind of grammar, a *regular* grammar, strings are derived from productions (rewriting rules) of the forms $S \rightarrow aW$ and $S \rightarrow a$, where $S$ and $W$ are *nonterminal symbols*, which do not appear in the final string, and $a$ is a *terminal symbol*, which appears as a letter in the final string. Searls has shown base pairing in RNA can be described by a *context-free grammar* (CFG), a more powerful class of formal grammars than the regular grammar (see Section 2.1). CFGs are often used to define the syntax of programming languages. A CFG is more powerful than a regular grammar in

[4] The energetic cost of adding a base pair to an existing helix varies, since each of the 16 possibilities has a different value. For example, adding `G-C` on top of `A-U` will cause the helix to have a different total energy than adding `G-U`. Since these effects accumulate, small changes can greatly affect a helix's overall energy [Mia94].

that it permits a greater variety of productions, such as those of the forms $S \rightarrow WY$ and $S \rightarrow aWb$ (capital letters represent nonterminals; lowercase letters represent terminals). As described by Searls, precisely these additional types of productions are needed to model the base-pairing structure in RNA.[5] In particular, productions of the forms $S \rightarrow \texttt{A}\ W\ \texttt{U}$, $S \rightarrow \texttt{U}\ W\ \texttt{A}$, $S \rightarrow \texttt{G}\ W\ \texttt{C}$ and $S \rightarrow \texttt{C}\ W\ \texttt{G}$ describe the structure in RNA due to Watson–Crick base pairing. Using productions of this type, a CFG can specify the language of biological palindromes.

Searls' original work [Sea92] argues the benefits of using CFGs as models for RNA folding, but does not discuss stochastic grammars or methods for creating the grammar from training sequences. Sakakibara *et al.* have provided an effective method for building a stochastic context-free grammar (SCFG) to model a family of RNA sequences [SBH$^+$93]; exactly this method is applied in this thesis to model spliceosomal small nuclear RNA sequences. (These are called *small* because their lengths are short relative to other forms of RNA—though they are on average two to three times as long as tRNA sequences, on which the SCFG method described here was tested previously [SBU$^+$93, SBM$^+$94, SBH$^+$93]—and *nuclear* because they appear in cell nuclei.) Some analogues of stochastic grammars and training methods do appear in Searls' most recent work in the form of costs and other trainable parameters used during parsing [Sea93a, Sea93b, SD93]. but the integrated probabilistic framework discussed here, wherein probabilities are assigned to grammar productions which capture the possible base-pairing interactions, may prove to be simpler and more effective [SBH$^+$93, SBM$^+$94]. This method is closely related to the "covariance models" (CMs) of Eddy and Durbin [ED94]. CMs are equivalent to SCFGs but the algorithms for training and producing multiple alignments differ. An in-depth comparison of the two methods is given elsewhere [SBH$^+$94a].

This thesis describes the algorithm developed by Sakakibara *et al.* which was used to train SCFGs to model tRNA [SBH$^+$93, SBM$^+$94]. The algorithm, Tree-Grammar EM, deduces an SCFG's probabilities (parameters) automatically from a set of *unaligned* primary sequences with a novel generalization of the *forward-backward algorithm* commonly used to train HMMs. Tree-Grammar EM is based on tree grammars and is more efficient than the *inside-outside algorithm* [LY90], a computationally expensive generalization of the forward-backward algorithm developed to train SCFGs [Bak79]. Here, Tree-Grammar EM is used to derive, from small training sets of snRNA sequences (16 to 25 sequences per training set), three *trained grammars*. The training and testing sequences were taken from a 1993 database of aligned spliceosomal snRNAs [GP88, GRM93] maintained by Christine Guthrie *et al.* The alignments in this compilation are referred to as *trusted* alignments. Sequences in the three data sets ranged 53–170 bases in length.[6]

Specifying a probability for each production in a grammar yields a *stochastic* grammar. A stochastic grammar assigns a probability to each string it derives. Stochastic *regular*

[5] Although CFGs can not describe all RNA structure, they can account for enough to make useful models (e.g., for tRNA [SBH$^+$93]). CFGs cannot account for pseudoknots, structures generated when a single-stranded loop region base pairs with a complementary sequence outside the loop [tPD92, WPT89, Ple90]. Similarly, base triples involving three positions, as well as interactions in parallel (versus the more usual anti-parallel), are not currently modeled.

[6] The U2 and U4 sets were not modeled because their sequences were too long (96–194 and 120–162 bases in length, respectively) and their grammars were too large (2356 and 2221 productions, respectively); the local implementation of the Tree-Grammar EM parsing step required more core memory than was available in local computers.

grammars are exactly equivalent to HMMs and suggest an interesting generalization from HMMs to stochastic *context-free* grammars [Bak79].

This thesis discusses stochastic models for spliceosomal small nuclear RNAs developed using SCFGs as in previous work by Sakakibara *et al.* [SBH$^+$93]. These models resemble the protein HMMs of Krogh *et al.* [KBM$^+$94] but they incorporate base-pairing information. The process used here for building an SCFG that forms a statistical model of a subset of spliceosomal small nuclear RNA sequences is similar to the process employed by Krogh *et al.* to construct an HMM representing a statistical model of a protein family. Each model is used to discriminate its set from other RNAs of similar length, and to obtain a multiple alignment for each sequence set (training plus test) in the same manner as for proteins. Also, each model is employed to determine the base pairing that defines their secondary structure of snRNAs for which only the primary sequence is known.

This thesis focuses on the ribonucleic acid (RNA) portions of the spliceosome. The spliceosome includes three major RNA–protein subunits, the so-called `U1`, `U2` and `U4/U6.U5` *small nuclear ribonucleoprotein (snRNP)* particles, as well as an additional group of non-snRNP protein splicing factors. Each of the snRNP spliceosomal subunits contains one or more *small nuclear RNAs (snRNAs)* and a set of snRNP proteins. This thesis examines only the `U1`, `U5` and `U6` snRNA sequences in these subunits. `U6` is the most highly conserved of the spliceosomal snRNAs. As implied by the clustered name "`U4/U6.U5`" for one of the three major subunits, `U4` and `U6` are usually involved in intermolecular base pairing (that is, regions of `U4` base pair with regions of `U6`).

To understand the role of the spliceosome, it is useful to summarize protein synthesis, the process of mapping from a DNA sequence to a folded protein: Part of a DNA strand, a gene, is *transcribed* to a messenger RNA (mRNA) strand, which is then *translated* to protein. DNA and RNA have four-letter alphabets, one for each ribonucleic acid, while proteins have a 20-letter alphabet, one for each amino acid. Specific subsequences of three letters in mRNA, called *codons*, specify single amino acids. Each RNA letter represents a *nucleotide* or *base*, which consists of three types of molecules bound together (a sugar, a phosphate and a *purine* [`A` or `G`] or *pyrimidine* [`C` or `U`]).

Although not involved directly in protein synthesis, the spliceosome provides a required function, pre-mRNA splicing, which enables protein synthesis to take place and which is also important for gene expression and biological regulatory mechanisms. In eucaryotes, the spliceosome catalyzes the removal of *introns* from pre-mRNA. That is, it splices introns (intervening regions in genes) out of mRNA in the nucleus of a eukaryotic cell and conjoins the remaining *exons* (expressed regions) to produce a so-called mature mRNA. In the context of protein synthesis, the spliceosome needs to have done this before the mature mRNA can leave the cell nucleus to bind to a ribosome in the cytoplasm, where it is translated into a protein [Hun92]. As Guthrie and Patterson note, "Presumably the low information content within pre-mRNA introns is compensated by the participation of the snRNPs, which impart the appropriate structure for catalysis" [GP88].

An RNA sequence has a distinct orientation determined by its chemical backbone, a string of alternating sugar and phosphate molecules. This orientation is indicated by a label on each end: $5'$ indicates the phosphate-terminated end of the sequence (phosphate attaches to the $5'$ carbon of a ribose sugar), while $3'$ indicates the sugar-terminated end [Cur79]. Some nucleotides may be modified by the addition of small chemical groups, but in this work, as in other computer approaches for modeling biological sequences, the modified nucleotides are converted to their unmodified forms.

Using stochastic context-free grammars, I hope to elucidate secondary structures for snRNA sequences, both for those with trusted alignments available and for those with unknown alignments. These new secondary structures may help biologists to determine and to better understand these molecules' functions. Concomitantly, I desire to demonstrate the usefulness of SCFGs for folding and aligning RNA molecules that are longer and more complex than the transfer RNA (tRNA) molecules previously used to test this method.

Results show that after having been trained on as few as about 20 snRNA sequences, each of the three models can discern snRNA sequences from similar-length RNA sequences of other kinds, can find secondary structure of new snRNA sequences and can produce multiple alignments of snRNA sequences.

## 1.2 Thesis structure

Chapter 2 presents the methods and training experiments. Chapter 3 describes the experimental results. Chapter 4 discusses possible future tasks.

# 2. Methods

The Tree-Grammar EM algorithm is used to produce three stochastic context-free grammars, one for each of the three snRNA training sets. This chapter explains how these grammars are created and used, and how the Tree-Grammar EM algorithm works.

## 2.1 Context-free grammars for RNA

A grammar is principally a set of productions (rewrite rules) that is used to generate a set of strings, a *language*. The productions are applied iteratively to generate a string, a process called *derivation*. For example, application of the productions in Figure 2.1 could generate the RNA sequence `CAUCAGGGAAGAUCUCUUG` by the following derivation:

Beginning with the start symbol $S_0$, any production with $S_0$ left of the arrow can be chosen to replace $S_0$. If the production $S_0 \rightarrow S_1$ is selected (in this case, this is the only production available), then the symbol $S_1$ replaces $S_0$. This derivation step is written $S_0 \Rightarrow S_1$, where the double arrow signifies application of a production. Next, if the production $S_1 \rightarrow \texttt{C}\ S_2\ \texttt{G}$ is selected, the derivation step is $S_1 \Rightarrow \texttt{C}\ S_2\ \texttt{G}$. Continuing with similar derivation steps, each time choosing a nonterminal symbol and replacing it with the right-hand side of an appropriate production, the following derivation is obtained, terminating with the desired sequence:

$$\begin{aligned}
S_0 &\Rightarrow S_1 \Rightarrow \texttt{C}S_2\texttt{G} \Rightarrow \texttt{CA}S_3\texttt{UG} \Rightarrow \texttt{CA}S_4S_9\texttt{UG} \\
&\Rightarrow \texttt{CAU}S_5\texttt{A}S_9\texttt{UG} \Rightarrow \texttt{CAUC}S_6\texttt{GA}S_9\texttt{UG} \\
&\Rightarrow \texttt{CAUCA}S_7\texttt{GA}S_9\texttt{UG} \Rightarrow \texttt{CAUCAG}S_8\texttt{GA}S_9\texttt{UG} \\
&\Rightarrow \texttt{CAUCAGGGA}S_9\texttt{UG} \Rightarrow \texttt{CAUCAGGGAA}S_{10}\texttt{UUG} \\
&\Rightarrow \texttt{CAUCAGGGAAG}S_{11}\texttt{CUUG} \\
&\Rightarrow \texttt{CAUCAGGGAAGA}S_{12}\texttt{UCUUG} \\
&\Rightarrow \texttt{CAUCAGGGAAGAU}S_{13}\texttt{UCUUG} \\
&\Rightarrow \texttt{CAUCAGGGAAGAUCUCUUG}.
\end{aligned}$$

$$P = \{ \quad \begin{array}{llll}
S_0 \rightarrow S_1, & & S_7 \rightarrow \texttt{G}\ S_8, & \\
S_1 \rightarrow \texttt{C}\ S_2\ \texttt{G}, & & S_8 \rightarrow \texttt{G}, & \\
S_1 \rightarrow \texttt{A}\ S_2\ \texttt{U}, & & S_8 \rightarrow \texttt{U}, & \\
S_2 \rightarrow \texttt{A}\ S_3\ \texttt{U}, & & S_9 \rightarrow \texttt{A}\ S_{10}\ \texttt{U}, & \\
S_3 \rightarrow S_4\ \ S_9, & & S_{10} \rightarrow \texttt{C}\ S_{10}\ \texttt{G}, & \\
S_4 \rightarrow \texttt{U}\ S_5\ \texttt{A}, & & S_{10} \rightarrow \texttt{G}\ S_{11}\ \texttt{C}, & \\
S_5 \rightarrow \texttt{C}\ S_6\ \texttt{G}, & & S_{11} \rightarrow \texttt{A}\ S_{12}\ \texttt{U}, & \\
S_6 \rightarrow \texttt{A}\ S_7, & & S_{12} \rightarrow \texttt{U}\ S_{13}, & \\
S_7 \rightarrow \texttt{U}\ S_7, & & S_{13} \rightarrow \texttt{C} & \}
\end{array}$$

Figure 2.1: This set of productions $P$ generates RNA sequences with a certain restricted structure. $S_0, S_1, \ldots, S_{13}$ are nonterminals; `A, U, G` and `C` are terminals representing the four nucleotides.

A derivation can be arranged in a tree structure called a *parse tree* (Figure 2.2, left). A parse tree represents the syntactic structure of a sequence produced by a grammar. For an RNA sequence, this syntactic structure corresponds to a possible physical secondary structure (Figure 2.2, right). In general, multiple derivations (parse trees) are possible for a single sequence, providing several candidates for the secondary structure of that sequence.

Recent work in modeling RNA [SBU$^+$93, SBH$^+$93] uses context-free grammars having productions of the following forms: $S \rightarrow WY$, $S \rightarrow aWb$, $S \rightarrow aW$, $S \rightarrow aS$, $S \rightarrow W$ and $S \rightarrow a$, where $S$, $W$ and $Y$ are nonterminals and $a$ and $b$ are terminals. $S \rightarrow aWb$ productions describe the base pairings in RNA. $S \rightarrow aW$ and $S \rightarrow aS$ describe unpaired bases (loops). $S \rightarrow WY$ describe branched secondary structures. $S \rightarrow W$ are used to insert spaces in sequences to produce a multiple alignment. Since the snRNA SCFGs in this work generalize the protein HMMs used in previous work at UC Santa Cruz [HKMS93, KBM$^+$94], the three main types of snRNA SCFG nonterminals correspond to each of the primary states in a protein HMM: match, insert and skip. The *match nonterminals* in a grammar—the nonterminals on the left side of $S \rightarrow a$, $S \rightarrow aWb$ and $S \rightarrow aW$ types of productions—correspond to "important" positions in an RNA molecule (or columns in a multiple alignment). *Insert nonterminals*—the nonterminals on the left side of $S \rightarrow aS$ types of productions—also generate nucleotides but with different distributions. These are used to model loops by inserting nucleotides between important (match) positions. *Skip nonterminals*—nonterminals on the right of $S \rightarrow W$ productions—are used to skip a match nonterminal in a sequence derivation, so that no nucleotide appears at the corresponding position in a multiple alignment. In an SCFG modeling RNA, use of a skip production in parsing a sequence is equivalent to choice of a delete state in aligning a protein sequence to an HMM.

Formally, a context-free grammar $G$ consists of a set of nonterminal symbols $N$, a terminals alphabet $\Sigma$, a set of productions $P$, and the start symbol $S_0$. For a nonempty set of symbols $X$, let $X^*$ denote the set of all finite strings of symbols in $X$. Every CFG production has the form $S \rightarrow \alpha$ where $S \in N$ and $\alpha \in (N \cup \Sigma)^*$, thus the left-hand side consists of one nonterminal and there is no restriction on the number or placement of nonterminals and terminals on the right-hand side. The production $S \rightarrow \alpha$ means that the nonterminal $S$ can be replaced by the string $\alpha$. If $S \rightarrow \alpha$ is a production in $P$, then for any strings $\gamma$ and $\delta$ in $(N \cup \Sigma)^*$, the notation $\gamma S\delta \Rightarrow \gamma\alpha\delta$ is used to indicate that $\gamma S\delta$ *directly derives* $\gamma\alpha\delta$ in $G$. The string $\beta$ can be *derived* from $\alpha$, denoted $\alpha \overset{*}{\Rightarrow} \beta$, if there exists a sequence of direct derivations $\alpha_0 \Rightarrow \alpha_1$, $\alpha_1 \Rightarrow \alpha_2, \ldots, \alpha_{n-1} \Rightarrow \alpha_n$ such that $\alpha_0 = \alpha$, $\alpha_n = \beta$, $\alpha_i \in (N \cup \Sigma)^*$, and $n \geq 0$. Such a sequence is called a *derivation*. Thus, a derivation corresponds to an order of productions applied to generate a string. A particular derivation $d$ of the sequence $s$ using productions from the grammar $G$ is denoted $S_0 \overset{d}{\Rightarrow} s$. The grammar generates the language $\{w \in \Sigma^* \mid S_0 \overset{*}{\Rightarrow} w\}$, the set of all terminal strings $w$ that can be derived from the grammar.

## 2.2 Stochastic context-free grammars

In an SCFG, every production for a nonterminal $S$ has an associated probability value such that a probability distribution exists over the set of productions for $S$. (Any production with the nonterminal $S$ on the left side is called "a production for $S$.") Here the associated probability for a production $S \rightarrow \alpha$ in $G$ is denoted by $\mathcal{P}(S \rightarrow \alpha \mid G)$ or $\text{Prob}(S \rightarrow \alpha \mid G)$.

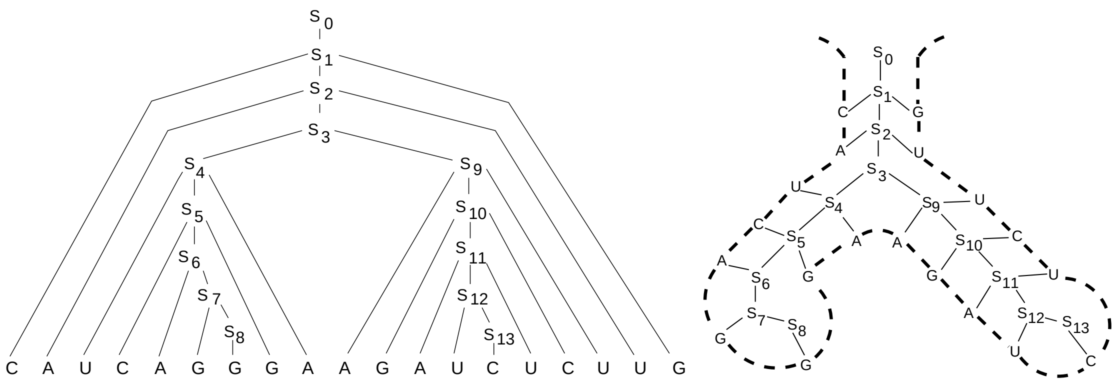


Figure 2.2: For the RNA sequence CAUCAGGGAAGAUCUCUUG, the grammar whose productions are given in Figure 2.1 yields this parse tree (left), which reflects a specific secondary structure (right).

Similarly, the probability for a derivation step $\alpha_i \Rightarrow \alpha_j$ given $G$ is denoted by $\text{Prob}(\alpha_i \Rightarrow \alpha_j \mid G)$.

A stochastic context-free grammar $G$ generates sequences and assigns a probability to each generated sequence, and hence defines a probability distribution on the set of sequences. The probability of a derivation (parse tree) can be calculated as the product of the probabilities of the production instances applied to produce the derivation. The probability of a sequence $s$ is the sum of probabilities over all possible derivations that $G$ could use to generate $s$:

$$\begin{aligned}\text{Prob}(s \mid G) &= \sum_{\substack{\text{all derivations} \\ \text{(parse trees) } d}} \text{Prob}(S_0 \overset{d}{\Rightarrow} s \mid G) \\ &= \sum_{\alpha_1, \ldots, \alpha_n} \text{Prob}(S_0 \Rightarrow \alpha_1 \mid G)\, \text{Prob}(\alpha_1 \Rightarrow \alpha_2 \mid G) \cdot \;\cdots\; \text{Prob}(\alpha_n \Rightarrow s \mid G)\end{aligned}$$

where terms $\alpha_i \in (N \cup \Sigma)^*$. For clarity, this work uses leftmost-first derivations, consistently replacing only the leftmost nonterminals to perform the derivation. If a sequence $s$ can be produced in multiple ways (that is, several derivations $S_0 \overset{d}{\Rightarrow} s$ exist), then the probabilities of those various ways must be summed.

Efficiently computing $\text{Prob}(s \mid G)$ presents a problem because the number of possible parse trees for $s$ is exponential in the length of the sequence. However, a dynamic programming technique analogous to the Cocke–Younger–Kasami or Early parsing methods [AU72] for non-stochastic CFGs can complete this task in polynomial time (specifically, in time proportional to the square of the number of nonterminals in the grammar $G$ times the average length of a typical sequence $s$, which in this work is on the order of $|s|^3$). The negative logarithm of the probability of a sequence given by the grammar $G$, $-\log(\text{Prob}(s \mid G))$, is defined as the *negative log likelihood (NLL) score* of the sequence. The NLL score quantifies how well the sequence $s$ fits the grammar—the likelihood that the grammar with its production probabilities could produce the sequence $s$.

CFGs have a drawback in that a sequence can sometimes be derived by a CFG in multiple ways. Since alternative parse trees reflect alternative secondary structures (foldings), a grammar may give several possible secondary structures for a single RNA sequence. An SCFG has the advantage that it can provide the most likely parse tree from this set of possibilities. If the productions are chosen carefully and the probabilities are estimated accurately, a parsing algorithm, when given grammar $G$ and an RNA sequence $s$, will produce a most likely parse tree for $s$ that corresponds to the correct secondary structure for $s$. The parser used in this work produces the single parse tree with the highest probability. It is possible that several parse trees may give one particular structure, and the sum of these trees' probabilities may be larger than the single most likely parse tree which gives another structure, but in this case the "correct" parse tree is taken to be the single latter parse tree. Indeed, for many of the snRNA sequences tested in this work, the most likely parse trees given by the corresponding snRNA-trained grammar match closely the accepted secondary structures, due to constraints inherent in the initial grammars and in the Tree-Grammar EM training procedure. The alternative parse trees may also be of interest because the same RNA sequence can adopt different structures (for example, some snRNA sequences), but these alternates are not considered in this work.

The most likely parse tree can be computed efficiently using a variant of the above procedure for calculating $\mathrm{Prob}(s \mid G)$. The most likely parse tree for the sequence $s$ can be obtained by calculating

$$\max_{\text{parse trees } d} \mathrm{Prob}(S_0 \overset{d}{\Rightarrow} s \mid G).$$

The dynamic-programming procedure to do this resembles the Viterbi algorithm for HMMs [Rab89] and takes time proportional to the square of the length of a typical sequence times the number of nonterminals in the grammar $G$. This procedure is also used to obtain multiple alignments: The parser aligns each sequence by finding the most likely parse tree given by the grammar, yielding an alignment of all nucleotides that correspond to the match nonterminals for each sequence, after which the mutual alignment of the sequences among themselves is determined. (Insertions of varying lengths can exist between match nonterminals, but by inserting enough spaces in each sequence to accommodate the longest insertion, an alignment of all the sequences is obtained.) This is equivalent to multiple alignment in a protein HMM, where the single most likely path for each sequence is computed.

## 2.3 Estimating SCFGs from sequences using Tree-Grammar EM

Both an SCFG's production probabilities and the productions themselves can in principle be chosen through examining an existing alignment of RNA sequences. Results using this approach to derive an SCFG to model tRNA were reported in previous work [SBU$^+$93].

At the other extreme, researchers recently have developed alternate methods for determining nearly all aspects of a grammar solely from training sequences [ED94]. To deduce a covariance model's structure (essentially, to choose an SCFG's productions), Durbin and Eddy use the standard Nussinov–Zuker dynamic-programming algorithm for RNA folding [NPGK78, Zuk89b], but with a non-standard cost function. Once a model structure exists, they train the model's parameters (production probabilities) using the Viterbi approximation of the expectation maximization (EM) algorithm.

In contrast, this thesis takes an intermediate approach. Prior information about snRNA structure is used to design appropriate initial grammars, but then training sequences are used to refine the estimates of the production probabilities in those grammars.

```
1. Start with an initial grammar $G_0$.
2. Use grammar $G_0$ and the $|s|^3$ CYK-like parsing algorithm to parse the raw input
sequences, producing a tree representation for each sequence indicating which nucleotides
are base-paired. This set of initial trees is denoted $T_0$. Set $T_{old} = \emptyset$ and $T_{new} = T_0$.
3. While $T_{new} \neq T_{old}$ do the following: {
    3a. Set $T_{old} = T_{new}$.
    3b. Use $T_{old}$ trees as input to the TG Reestimator algorithm, which iteratively
    re-estimates the grammar parameters until they stabilize. The grammar with
    the final stabilized probability values is called new grammar $G_{new}$.
    3c. Use grammar $G_{new}$ and the $|s|^3$ CYK-like parsing algorithm to parse the
    input sequences, producing a new set of trees $T_{new}$.
}
```

Figure 2.3: The Tree-Grammar EM training algorithm, shown in pseudocode here, performs the TG Reestimator and CYK-like parsing algorithms as substeps. In Step 3b, the probability values are considered "stabilized" when the difference between consecutive NLL scores for the sequences, computed using the current and the previous grammar parameters, has become smaller than a pre-chosen value called the *stopping criterion*. The smaller the stopping criterion, the larger the number of iterations within Step 3b. In this and previous tRNA work, the stopping criterion was set to 1.0 the first time through Step 3, then decreased to 0.1 subsequently, leading to, on average, about six iterations in Step 3b.

To estimate the SCFG parameters from unaligned training tRNA sequences, Sakakibara *et al.* introduced Tree-Grammar EM (Figure 2.3), a new method for training SCFGs that uses a generalization of the forward-backward algorithm commonly used to train HMMs. This generalization, called *TG Reestimator*, is more efficient than the inside-outside algorithm, which was previously proposed to train SCFGs.

The inside-outside algorithm [LY90, Bak79] is an *expectation maximization* (EM) algorithm that calculates maximum likelihood estimates of an SCFG's parameters based on training data. (*Expectation* refers to the calculation of an auxiliary function that depends on the current and the reestimated model, while *maximization* refers to maximization over the reestimated model [Rab89].) However, it requires the grammar to be in Chomsky normal form, which is possible but inconvenient for modeling RNA (and requires more nonterminals). Further, it takes time at least proportional to $|s|^3$ per training sequence $s$, whereas the forward-backward procedure for HMMs takes time proportional to $|s|$ per training sequence, where $|s|$ is the length of sequence $s$. In addition, the inside-outside algorithm is prone to settling in local minima; this presents a problem when the initial grammar is not highly constrained.

To avoid these problems, Sakakibara *et al.* developed the iterative algorithm TG Reestimator (Step 3b in Figure 2.3). While the running times of both Tree-Grammar EM and the inside-outside algorithm are asymptotically equivalent due to the use of the $|s|^3$ CYK-like algorithm (Step 3c of Figure 2.3), in practice Tree-Grammar EM is more efficient. In Tree-Grammar EM, the inner loop (Step 3b) takes time proportional to $|s|$ per sequence per iteration (the grammar size is constant), while in the inside-outside algorithm, the analogous step takes time proportional to $|s|^3$ per sequence per iteration. The difference is that the inside-outside algorithm computes values for all possible parses for *all possible* tree structures for the sequence $s$, whereas the Tree-Grammar EM method computes all possible parses for only the *current* tree structure provided for the sequence $s$. Since the number of iterations in Step 3b is typically on the order of 10, while the number of iterations of Step 3 is typically four or five, Tree-Grammar EM is more practical for longer RNA sequences.

TG Reestimator requires folded RNA sequences as training examples, rather than unfolded ones. Thus, some tentative "base pairs" in each training sequence have to be identified before TG Reestimator can begin. The procedure to do this involves designing a rough initial grammar (see Section 2.5) that may represent only a portion of the base-pairing interactions (Step 1) and parsing the unfolded RNA training sequences to obtain a set of partially folded RNA sequences (Step 2). Then a new SCFG can be estimated using the partially folded sequences and TG Reestimator (Step 3b). Further productions might be added to the grammar at this step, though this thesis does not explore this approach. The parameter re-estimation is then repeated. In this way, TG Reestimator can be used even when precise knowledge of the base pairing is not available. TG Reestimator constitutes one part of the entire training procedure, Tree-Grammar EM.

The Tree-Grammar EM procedure is based on the theory of stochastic tree grammars [TW68, Fu82, Sak92]. Tree grammars are used to derive labeled trees instead of strings. Labeled trees can be used to represent the secondary structure of RNA easily [Sha88, SZ90] (see Figure 2.2). A tree grammar for RNA denotes both the primary sequence and the secondary structure of each molecule. Since these are given explicitly in each training molecule, the TG Reestimator algorithm does not have to (implicitly) sum over *all* possible interpretations of the secondary structure of the training examples when re-estimating the grammar parameters, as the inside-outside method must do. The TG Reestimator algorithm iteratively finds the best parse for each molecule in the training set and then readjusts the production probabilities to maximize the probability of these parses. The new algorithm also tends to converge faster because each training example is more informative [SBU+93].

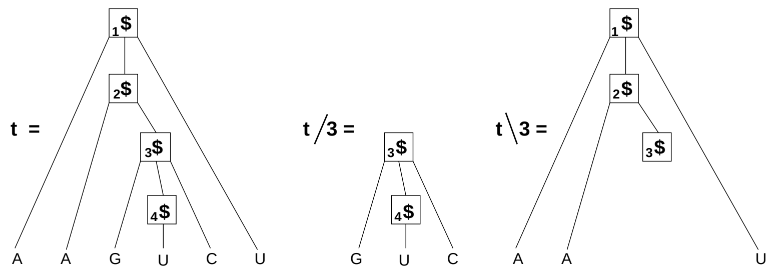


Figure 2.4: The folded RNA sequence (AA(GUC)U) can be represented as a tree $t$ (left), which can be broken into two parts such as $t/3$ (middle) and $t\backslash 3$ (right). The \$ symbols act as placeholders for nonterminal labels on internal nodes. The root (node 1) and the third internal node (node 3) represent A-U and G-C base pairs, respectively, while the other nodes (2 and 4) indicate unpaired nucleotides.

Before the Tree-Grammar EM algorithm can be described, some tree definitions are needed: A *tree* is a rooted, directed, connected acyclic finite graph in which the direct successors of any node are linearly ordered from left to right. The predecessor of a node is called the *parent*; the successor, a *child*; and a child of the parent, a *sibling*. In this work, a folded RNA sequence $s$ is represented by a *labeled tree* $t$ as follows. Each leaf node is labeled by one of four nucleotides {`A`, `U`, `G`, `C`}. Every internal node is labeled by a nonterminal. The sequence of nucleotides labeled at leaf nodes traced from left to right exactly constitutes the RNA sequence $s$, and the structure of the tree represents its folding structure.

Figure 2.4 shows a tree representation of the folded RNA sequence `(AA(GUC)U)`, where dollar signs \$ act as a placeholders for the nonterminals labeling the internal nodes. The parsing step of Tree-Grammar EM uses the current grammar to choose which actual nonterminals should label these internal nodes.

Assume all internal nodes in $t$ are numbered from 1 to $T$ (the number of internal nodes) in some order. Then, in standard tree-grammar notation, for an internal node $n$ $(1 \le n \le T)$, $t/n$ denotes the subtree of $t$ with root $n$ (Figure 2.4, center) and $t \backslash n$ denotes the tree obtained by removing a subtree $t/n$ from $t$ (Figure 2.4, right).

The probability of any folded sequence $t$ given by an SCFG $G = (N, \Sigma, P, S_0)$ is calculated efficiently using a dynamic-programming technique, as is done with the forward algorithm in HMMs. A labeled tree $t$ representing a folded RNA sequence has the shape of a parse tree, so to parse the folded RNA, the grammar $G$ needs only to label each internal node with an appropriate nonterminal according to the productions. For all nonterminals $S$ and all nodes $n$ such that $1 \le n \le T$, let $in_n(S)$ be the probability that the subtree $t/n$ can be derived given that the nonterminal $S$ is assigned to node $n$ and given grammar $G$. Then $in_n(S)$ can be calculated inductively as follows:

**1.** Initialization: $$in_n(a) = \begin{cases} 1 & \text{if } a \text{ is the nucleotide appearing at leaf node } n, \\ 0 & \text{otherwise,} \end{cases}$$

for all leaf nodes $n$ and terminals $a$ (nucleotides). This extension of $in_n(S)$ is for the convenience of the inductive calculation of $in_n(S)$.

**2.** Induction:

$$in_m(S) = \sum_{\substack{Y_1, \ldots, Y_k \\ \in (N \cup \Sigma)}} in_{n_1}(Y_1) \ \cdots \ in_{n_k}(Y_k) \cdot \ \mathcal{P}(S \to Y_1 \ \cdots \ Y_k),$$

for all nonterminals $S$, all internal nodes $m$ and all $m$'s children nodes $n_1, \ldots, n_k$.

**3.** Termination: For the root node $n$ and the start symbol $S_0$,

$$\text{Prob}(t \mid G) \ = \ in_n(S_0). \tag{2.1}$$

Intuitively, if $s'$ represents the substring of $s$ found at the leaves of the subtree rooted at the internal node $n$ inside the full parse tree $t$, then $in_n(S)$ represents the probability that the nonterminal $S$ can derive that substring $s'$ when a production for $S$ is applied in a derivation step at node $n$. For example, in the tree $t$ shown in Figure 2.4, if $s' =$ `GUC` and if nonterminal $S$ is assigned to the third node such that $n = 3$, then $in_3(S)$ represents the probability that the nonterminal $S$ can derive the substring `GUC` via application of a production for $S$ in a derivation step at node 3. Thus, $in_n(S_0)$, where $n$ is $t$'s root node, is exactly the probability that the start nonterminal can derive the string $s$.

One more quantity is needed: $out_n(S)$ defines the probability of $t \backslash n$ given that the nonterminal $S$ is assigned to node $n$ and given grammar $G$. That is, $out_n(S)$ is the probability of $t \backslash n$, which is the tree $t$ excluding the subtree rooted at node $n$. This quantity is obtained similarly.

**1.** Initialization: For the root node $n$,

$$out_n(S) = \begin{cases} 1 & \text{for } S = S_0 \text{ (start symbol)}, \\ 0 & \text{otherwise.} \end{cases}$$

**2.** Induction:

$$out_m(S) = \sum_{\substack{Y_1,\ldots,Y_k \\ \in (N \cup \Sigma), \\ S' \in N}} in_{n_1}(Y_1) \cdots in_{n_k}(Y_k) \cdot \mathcal{P}(S' \to Y_1 \cdots S \cdots Y_k) \cdot out_l(S'),$$

for all nonterminals $S$, all internal nodes $l$ and $m$ such that $l$ is $m$'s parent and all nodes $n_1, \ldots, n_k$ are $m$'s siblings. Incorporated in this calculation are the probabilities of all productions for all nonterminals $S'$ that can derive the nonterminal $S$—namely, the productions $S' \to Y_1 \cdots S \cdots Y_k$. There is no termination step given in this case because the calculation of $\text{Prob}(t \mid G)$ is given in the termination step for $in_n(S)$.

Intuitively, $out_n(S)$ is the probability that the start nonterminal $S_0$ can generate $\alpha S \beta$, with nonterminal $S$ labeling node $n$, where subsequences $\alpha$ and $\beta$ result from excising the subsequence rooted at node $n$ from the full sequence $s$. Continuing the example from Figure 2.4, $out_3(S)$ is the probability of the partial derivation from the start symbol $S_0$ to the string `AA` $S$ `G` with nonterminal $S$ labeling node 3.

To determine how well a grammar fits a set of folded training sequences $t(1), \ldots, t(n)$, the probability that the grammar generates them needs to be calculated. "How well a grammar fits a set of folded sequences" means how likely it is that the grammar would predict exactly those foldings when presented with those sequences *unfolded*. This probability is simply a product of terms $\text{Prob}(t(j) \mid \text{G})$ as given by Equation 2.1, i.e.,

$$\text{Prob}(\text{sequences} \mid G) = \prod_{j=1}^{n} \text{Prob}(t(j) \mid G). \tag{2.2}$$

The goal is to obtain a high value for this probability, called the *likelihood* of the grammar. The *maximum likelihood* method of model estimation finds the model that maximizes the likelihood in Equation 2.2. There is no known way to directly and efficiently calculate the best model (the one that maximizes the likelihood) and avoid getting caught in suboptimal solutions during the search. However, the general EM method, given an arbitrary starting point, finds a local maximum by iteratively re-estimating the model such that the likelihood increases in each iteration, and often produces a solution that is acceptable, if perhaps not optimal. This method is often used in statistics.

Thus in Step 1 of Tree-Grammar EM (Figure 2.3), an initial grammar $G_0$ is created by assigning values to the production probabilities $\mathcal{P}(S \to Y_1 \cdots Y_k)$ for all $S$ and all $Y_1, \ldots, Y_k$, where $S$ is a nonterminal and $Y_i$ $(1 \leq i \leq k)$ is a nonterminal or terminal. If some constraints or features present in the sequences' multiple alignment are known, these are encoded in the initial grammar (see Section 2.5). The current grammar is set to this initial grammar.

In Step 3b of Tree-Grammar EM, using the current grammar, the values $in_n(S)$ and $out_n(S)$ for each nonterminal $S$ and each node $n$ for each folded training sequence are calculated in order to get a new estimate of each production probability, $\hat{\mathcal{P}}(S \to Y_1 \cdots Y_k)=$

$$\frac{\sum_{\text{all } t}\left(\sum_{\text{all } m} out_m(S)\cdot\mathcal{P}(S\rightarrow Y_1\cdots Y_k)\cdot in_{n_1}(Y_1)\cdots in_{n_k}(Y_k)/\text{Prob}(t\mid G)\right)}{\text{norm}},$$

which is a double sum over all sequences $t$ and all nodes $m$, where $G$ is the old grammar and "norm" is the appropriate normalizing constant such that $\sum_{Y_1,\ldots,Y_k}\hat{\mathcal{P}}(S\rightarrow Y_1\ \cdots\ Y_k)=1$. A new current grammar $G_{new}$ is created by replacing all $\mathcal{P}(S\rightarrow Y_1\cdots Y_k)$ with the re-estimated probabilities $\hat{\mathcal{P}}(S\rightarrow Y_1\cdots Y_k)$.

## 2.4 Overfitting and regularization

Attempts to estimate a grammar with too many free parameters from a small set of training sequences will encounter the *overfitting* problem—that is, the grammar fits the training sequences well, but poorly fits other, related (test) sequences. One solution is to use *regularization* to control the effective number of free parameters. Following the method used in recent SCFG modeling of tRNA [SBU+93, SBM+94, SBH+93, SBH+94a], this work regularizes the grammars taking a Bayesian approach to the parameter estimation problem, similar to a previous approach taken with protein HMMs [KBM+94, BHK+93].

Before the grammars are trained, a prior probability density is constructed for each of their "important" parameter sets: $S\rightarrow aWb$ productions and $S\rightarrow aW$ productions, where $S$ and $W$ are nonterminals and terminal symbols $a,b$ are drawn from the set of four nucleotides $\{\texttt{A},\texttt{C},\texttt{G},\texttt{U}\}$. This prior probability density takes the form of a single-component Dirichlet distribution [SD89].

The $S\rightarrow aWb$ productions, which generate base pairs, come in groups of 16, corresponding to all possible pairs of terminal symbols. The $S\rightarrow aW$ productions, which generate nucleotides in loop regions, come in groups of four. For the base-pairing productions, the Dirichlet prior information about which productions are most likely is employed. For instance, Watson–Crick pairs are more frequently observed than other base pairs. Using the large alignment of 16S rRNA sequences [LOM+93], the method described by Brown *et al.* [BHK+93] was used to obtain the 16 parameters of a Dirichlet density over possible base-paired position distributions. These probabilities were used to calculate precise prior information about base-pair probabilities. Similarly the 16S rRNA alignment was used to calculate a four-parameter Dirichlet prior for nucleotide distributions in loop match positions. Further details, with references to protein HMMs, are presented in the paper by Brown *et al.* [BHK+93].

These parameters constitute the regularizer (Figure 2.5). The small parameter values imply that the 16S rRNA data set from which they were computed is highly variable. The values do not sum to one because they are not probabilities; normalizing the Dirichlet parameters will, however, yield the average distribution that the Dirichlet prior specifies. The parameters are added as "pseudocounts" during each re-estimation step of Tree-Grammar EM (step 3b in Figure 2.3). Thus, at each iteration, TG Reestimator computes mean posterior estimates of the model parameters rather than maximum likelihood estimates.

Regularization is performed in a similar manner for probability distributions for other production types, including chain rules $S\rightarrow W$, branch productions $S\rightarrow WY$ and insert productions $S\rightarrow aS$. Insert productions (for loops) are regularized with very large uniform pseudocounts over the four possible nucleotides so that their probability distributions will be fixed at uniform values rather than estimated from the training data. This is equivalent

| $S \rightarrow aWb$ | | 3′ | | | |
|---|---|---|---|---|---|
| | | C | G | U | A |
| | C | 0.134879 | 3.403940 | 0.162931 | 0.176532 |
| 5′ | G | 1.718997 | 0.246768 | 0.533199 | 0.219045 |
| | U | 0.152039 | 0.784135 | 0.249152 | 2.615720 |
| | A | 0.135167 | 0.192695 | 1.590683 | 0.160097 |

| $S \rightarrow aW$ | C | G | U | A |
|---|---|---|---|---|
| | 0.21 | 0.18 | 0.20 | 0.26 |

Figure 2.5: Helix (top) and loop (bottom) pseudocounts are added to actual observed frequencies to reflect prior information. These counts are based upon estimated Dirichlet distributions for helix regions and loop regions. The matrix is asymmetric because the distributions differ with the base ordering in a base pair (ex., 5′ C paired with 3′ G has higher probability than 5′ G paired with 3′ C).

| Grammar | Number of Productions | Number of Nonterminals |
|---|---|---|
| U1 | 2021 | 462 |
| U5 | 1498 | 367 |
| U6 | 1761 | 498 |

Figure 2.6: Shown are the total number of productions and nonterminals for the grammars that model the three snRNA sets.

to the regularization used previously for the insert states of protein HMMs [KBM+94]. This further reduces the number of parameters to be estimated, helping to avoid overfitting.

## 2.5 The initial grammars

The initial grammars were based on snRNA structures previously described [GP88]. The lengths of helices and loops were approximated via empirical examination of the provided trusted alignments. Specifically, the most common substructure lengths were incorporated directly as the number of match nonterminals to appear in the actual grammar for those substructures. Though the number of match nonterminals was determined this way, insertion and skip productions for the same nonterminals are included to accommodate longer loops, or shorter loops or helices.

The grammars were formulated as *meta-grammars* which were translated into actual grammars with appropriate productions, nonterminals and terminals. (The actual grammars vary in size, as tabulated in Figure 2.6. The U5 grammar was the smallest, and thus took the least time to train.) Each meta-grammar has *meta-nonterminals* such as `trunkHelix` and `loop3` corresponding to snRNA structures. Each of these meta-nonterminals has a set of actual productions associated with it (not shown here). Sakakibara *et al.* wrote a program that automatically generates actual productions given only the meta-grammar. To further simplify grammar specification, recently Leslie Grate wrote a graphical program called `scfgedit`, which allows users to define meta-grammars easily.

The three types of meta-nonterminals are BRANCH, LOOP and HELIX. Each LOOP or HELIX meta-nonterminal has an associated length, given by a numeric parameter. For a meta-nonterminal LOOP($\ell$), the grammar generating program creates a subgrammar that is equivalent to an HMM model with $\ell$ match states as described in previous work on proteins [KBM+94], except that the four-letter nucleic-acids alphabet replaces the twenty-letter amino-acids alphabet. Distributions of the nucleotides in such a loop are defined by the probabilities of the productions for $\ell$ match nonterminals. Longer or shorter loops can be derived using special nonterminals and productions that allow position-specific insertions and deletions. For a meta-nonterminal HELIX($\ell$), the grammar generating program creates a subgrammar consisting of $\ell$ nonterminals. Each nonterminal has 16 productions that derive possible base pairs for its position in the helix. Each nonterminal has its own probability distribution over these 16 possible productions. These probability distributions, like those for match nonterminals in loops, are initially defined using Dirichlet priors (Section 2.4). Other nonterminals and productions are added to allow deletions of base pairs, enabling helix length variations.

To form the complete actual grammar, all the subgrammars for the various structures are combined according to the high-level specification. Special treatment of nonterminals involved in branch productions of the form $S \rightarrow SS$ can also be included. In particular, the snRNA grammars were specified such that certain branch productions may also, with some probability, omit one of the nonterminals on the right-hand side. This allows the grammar to derive snRNAs that are missing certain substructures, such as arms or loops. In general, any substructure in the grammar can be specified to be absent with some probability. These probability values are initialized to default prior values and then re-estimated during training on actual sequences, as are all the production probabilities in the grammar.

Represented in three forms (*XRNA*, `scfgedit` and text) in Figures 2.8, 2.9 and 2.10 are the three initial grammars designed for the snRNA experiments. The first of the three forms is a graphical rendering generated using *XRNA*, an X Windows-based program for editing and displaying RNA primary, secondary and tertiary structure [WGN93]. Using simple filters, a secondary structure predicted by a grammar can be transformed into XRNA format. The `scfgedit` program outputs an XRNA representation of the grammar's match productions, with a single `G-C` placeholder representing each set of 16 actual base-pairing productions for a single helix-match nonterminal and a single `A` placeholder representing each set of four actual productions for a single loop-match nonterminal. Thus, all nucleotides in the XRNA grammar pictures are merely placeholders, showing only secondary structure (match positions) but not a true consensus structure. A black circle indicates the $5'$ end of each structure. The second format shown in the figures is `scfgedit` form, with the start meta-nonterminal $S$ circled. Squares represent helices, circles represent loops, and triangles represent branch points. The third format is the text output from `scfgedit` describing the meta-grammar.

Lengths chosen for each helix and loop appear in the upper left corner of each square or triangle in the `scfgedit` representations. These were selected to equal the most common lengths evident in the input data alignment for each data set. For example, if there are 10 sequences in a particular data set and they each have a helix that appears in the trusted alignment as follows,

```
...start-helix.loop.-end-helix-...
...-GCC-------.AAAG.-------GGC-...
...-GC--------.AAA-.--------GC-...
...-GC--------.AAA-.--------GC-...
```

| Grammar | Helix | Current Length | Better Length |
|---|---|---|---|
| U1 | `stem2` | 16 | 18 |
| U5 | `trunkBottom` | 4 | 6 |
| | `trunkTop` | 8 | 9 |
| | `lastHelix` | 7 | 10 |
| U6 | `5'stem` | 8 | 12 |

Figure 2.7: These length changes would be required to make each grammar "perfectly fit" the sequence set they are intended to model, such that predicted alignments would be able to perfectly match the trusted alignments. Without these changes, the predicted alignments of helices are strictly length-limited but the grammars are smaller and thus easier to train.

```
...-CC--------.AAAG.--------GG-...
...-CCGGCCCGGG.-AAC.CCCGGGCCGG-...
...-GC--------.-AAG.--------GC-...
...-G---------.AAAG.---------C-...
...-GC--------.-AAG.--------GC-...
...-CC--------.AAAC.--------GG-...
...-GC--------.AAA-.--------GC-...
```

then the length of the helix is set to two, the most common length of that helix. The length of the loop in this example is set to three. The sequences in which that helix is shorter than two bases are accounted for in the production probabilities for the delete productions for the match nonterminals defining the helix.

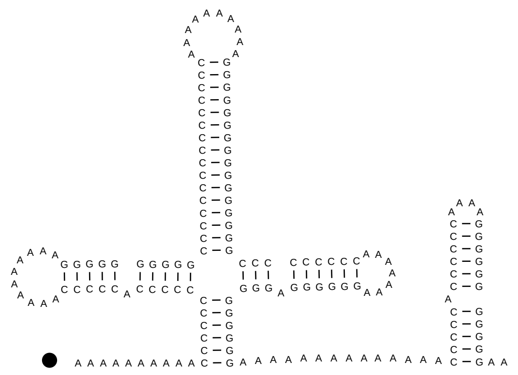
Sequences having that helix longer than two bases, such as the fifth sequence in the above example, are not modeled perfectly by the grammar because, for simplicity, the initial grammars do not include insert productions for helix-match nonterminals. Specifically, loops have both insert and delete nonterminals in addition to match nonterminals, such that a predicted loop alignment can be either longer or shorter than the number of loop match nonterminals, while helices have only delete nonterminals in addition to match nonterminals, such that a predicted helix alignment can be only as long as the number of helix-match nonterminals. This holds true also in previous tRNA work [SBU+93, SBM+94, SBH+93, SBH+94a].

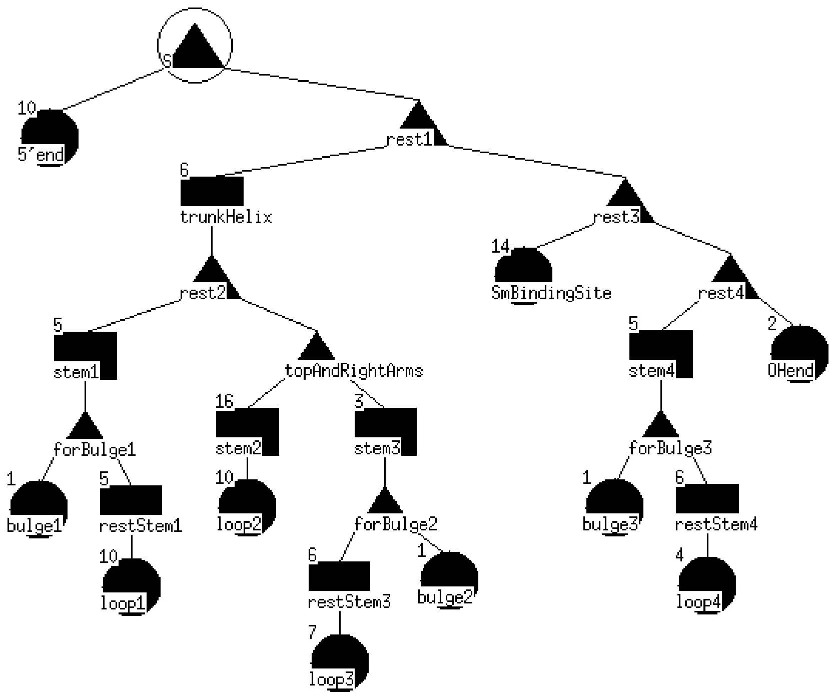
A "perfectly fitting" model can be obtained if the helix lengths are set to be the *longest* observed length in the trusted alignment (10, in the above helix example) rather than the *most common* observed length, but this makes the grammar larger and thus increases computation time for training and parsing. Using an imperfect model can restrict the ability of the predicted alignment to match the trusted when a helix's length varies widely from sequence to sequence in the modeled set, as is the case with the snRNAs; sequences having a longer length for that helix in the trusted alignment are not represented in the predicted alignment ("extra" bases are placed as part of a loop on one side or the other of the helix).

If the grammars of this work were altered to perfectly fit the sequence set they are intended to model, then helices would need to be lengthened as indicated in Figure 2.7. Making the helices longer in the grammar would not affect the alignment of sequences in which those helices are shorter than the maximum length, because the skip productions would allow insertion of spaces in the "extra" base-pairing match positions. However, making the helices longer would increase the grammar's numbers of nonterminals and productions, which would in turn enlarge the required time for grammar training and sequence parsing.

Currently the grammar-generating program does not generate special insert productions to allow for *bulges* (extra nucleotides inserted between two bases on one side of a helix) as in the grammars of Durbin and Eddy [ED94], though it is straightforward to add this ability. It would involve adding insert productions of the form $S \rightarrow aS$ and $S \rightarrow Sa$ for helix match nonterminals, such that a single nucleotide could be inserted between any two positions on one side of a helix. Thus, a sequence whose most likely parse uses these productions would have a bulge in its predicted alignment. Implementing this method would result in a more complicated grammar.

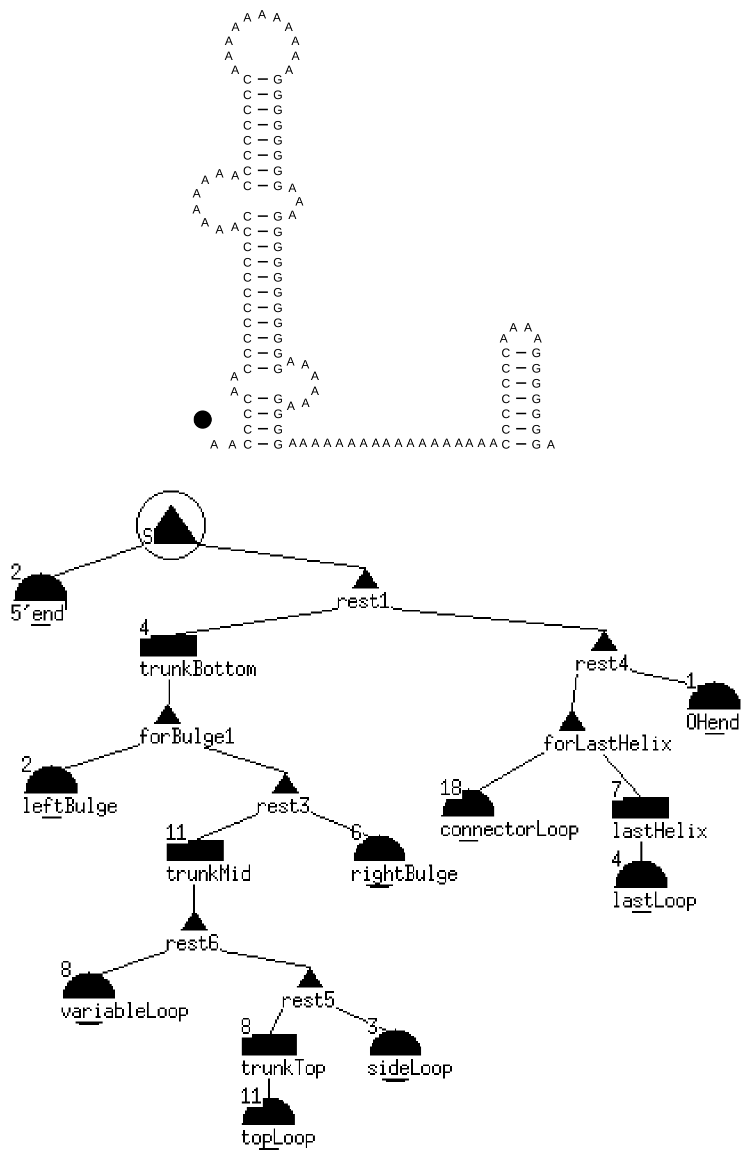
An attempt was made to model some of the bulges by translating a single helix containing a bulge indicated in a trusted alignment into two helices in series joined by a single strand. For example, Stem I in the `U1` grammar is divided into `stem1` of length 5 followed by `restStem1` of length 5, with the single strand `bulge1` connecting them. This does capture some structure, but at the expense of restricting the possible positioning of bulges in multiple alignments as well as making the grammars more complex. The grammars might be more efficient and flexible if bulges were modeled by introducing new productions into the grammar as mentioned in the previous paragraph.





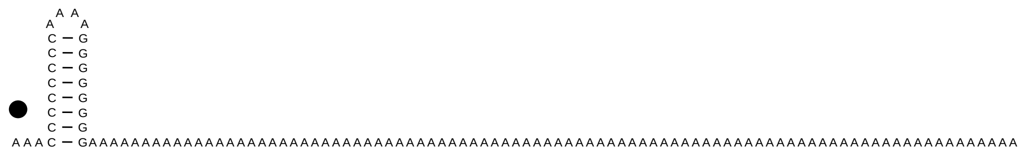
| | | | |
|---|---|---|---|
| S | 2BranchRNec | 5'end | rest1 |
| 5'end | Loop | 10 | |
| rest1 | 2BranchRNec | trunkHelix | rest3 |
| trunkHelix | Helix | 6 | rest2 |
| rest2 | 2BranchRNec | stem1 | top&Right |
| stem1 | Helix | 5 | forBulge1 |
| forBulge1 | 2BranchRNec | bulge1 | restStem1 |
| bulge1 | Loop | 1 | |
| restStem1 | Helix | 5 | loop1 |
| loop1 | Loop | 10 | |
| top&Right | 2BranchRNec | stem2 | stem3 |
| stem2 | Helix | 16 | loop2 |
| loop2 | Loop | 10 | |
| stem3 | Helix | 3 | forBulge2 |
| forBulge2 | 2BranchRNec | restS3 | bulge2 |
| restS3 | Helix | 6 | loop3 |
| loop3 | Loop | 7 | |
| bulge2 | Loop | 1 | |
| rest3 | 2BranchRNec | SmSite | rest4 |
| SmSite | Loop | 14 | |
| rest4 | 2BranchRNec | stem4 | OHend |
| stem4 | Helix | 5 | forBulge3 |
| forBulge3 | 2BranchRNec | bulge3 | restS4 |
| bulge3 | Loop | 1 | |
| restS4 | Helix | 6 | loop4 |
| loop4 | Loop | 4 | |
| OHend | Loop | 2 | |

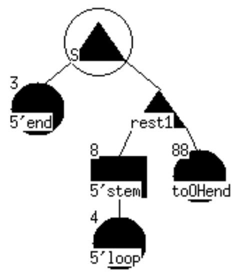
Figure 2.8: Represented in XRNA format (top), `scfgedit` format (middle) and text format (bottom) is the initial grammar for the `U1` set.



| | | | |
|---|---|---|---|
| S | 2BranchRNec | 5'end | rest1 |
| 5'end | Loop | 2 | |
| rest1 | 2BranchRNec | trunkBtm | rest4 |
| trunkBtm | Helix | 4 | forBulge1 |
| forBulge1 | 2BranchRNec | leftBulge | rest3 |
| leftBulge | Loop | 2 | |
| rest3 | 2BranchRNec | trunkMid | rightBlg |
| trunkMid | Helix | 11 | rest6 |
| rest6 | 2BranchRNec | varLoop | rest5 |
| varLoop | Loop | 8 | |
| rest5 | 2BranchRNec | trunkTop | sideLp |
| trunkTop | Helix | 8 | topLoop |
| topLoop | Loop | 11 | |
| sideLp | Loop | 3 | |
| rightBlg | Loop | 6 | |
| rest4 | 2BranchRNec | forLastHlx | 0Hend |
| forLastHlx | 2BranchRNec | connLp | lastHlx |
| connLp | Loop | 18 | |
| lastHlx | Helix | 7 | lastLoop |
| lastLoop | Loop | 4 | |
| 0Hend | Loop | 1 | |

Figure 2.9: Represented in XRNA format (top), `scfgedit` format (middle) and text format (bottom) is the initial grammar for the `U5` set.





```
S         2BranchRNec   5'end    rest1
5'end     Loop          3
rest1     2BranchRNec   5'stem   to0Hend
5'stem    Helix         8        5'loop
5'loop    Loop          4
to0Hend   Loop          88
```

Figure 2.10: Represented in XRNA format (top), `scfgedit` format (middle) and text format (bottom) is the initial grammar for the `U6` set.

# 3. Experimental Results

This chapter delineates in detail the application of Tree-Grammar EM, described in the previous chapter, to deduce three trained grammars from three distinct training sets of unfolded and unaligned snRNA sequences (Figure 3.3). The procedure follows that done for tRNA previously [SBU+93, SBM+94, SBH+94a].

| Grammar | Average Time to Train (Step 3b) | Average Time to Parse An snRNA | Average Time to Parse A Non-snRNA | Length of Sequences |
|---|---|---|---|---|
| `U1` | 5.1 hrs | 0.5 min | 0.4 min | 156 |
| `U5` | 1.0 hrs | 0.4 min | 0.3 min | 113 |
| `U6` | 13.2 hrs | 0.7 min | 0.6 min | 100 |

Figure 3.1: A Decstation 5000/240 with 128 Mbytes memory required these CPU times to perform Step 3b of Tree-Grammar EM in Figure 2.3 and to run the $|s|^3$ CYK-like parsing algorithm. Column 1 shows the time taken for the grammar parameters to stabilize in TG Reestimator. Columns 2 and 3 show times taken to parse two equal-length sequences using the trained grammars: an snRNA sequence from the modeled set, and a non-snRNA sequence of the same length.

Using initial grammars designed as discussed in Section 2.5, Tree-Grammar EM produced three trained grammars, one for each set. The initial grammars' production probabilities were established using only the Dirichlet pseudocount information. Tree-Grammar EM was used to refine these initial grammars with varying numbers of training sequences (Figure 3.3). Table 3.1 indicates the run times taken to train each grammar and parse raw sequences given the grammar on a Decstation 5000/240 with 128 Mbytes memory. During the training process, only the probabilities of the productions were reestimated and no nonterminals or productions were added or deleted, unlike "model surgery" in previous HMM work [KBM+94].

## 3.1 Data

The experiments for generating and testing the trained grammars used data from three sources:

**1.** The three snRNA training sets are from Christine Guthrie and Saira Mian [GRM93]. Some interpretation was performed to obtain a trusted multiple alignment for each set, as none of the helices was labeled, except for those in the `U5` set. This involved my examining the aligned sequences and comparing aligned substrings with substructures in the published consensus structures [GP88] to determine a probable labeling. (See Section 2.5.) These labelings were then confirmed to be correct by Saira Mian. The provided aligned sequences, along with their inferred labelings, are referred to as the *trusted* alignments.

The trusted alignments constitute the current best estimates of foldings that biologists have devised or deduced for the snRNA sets, but these can change over time as new sequences or correlations are discovered. These alignments in particular were produced by comparing each sequence to that of a single organism, the yeast *Saccharomyces cerevisiae*, and searching for motif structures in each sequence that are observed in that yeast sequence.

| Data Set | Original Number of Sequences | Number Sequences Removed Extra-long | Containing N | Number Remaining | Number Incomplete |
|---|---|---|---|---|---|
| U1 | 54 | 3 | 2 | 49 | 6 |
| U5 | 34 | 2 | 1 | 31 | 2 |
| U6 | 49 | 2 | 3 | 44 | 2 |
| Totals | 137 | 7 | 6 | 124 | 10 |

Figure 3.2: As shown, 7 extra-long snRNA sequences [GP88, GRM93] and 6 sequences containing the N character were omitted before sequences were partitioned into training and test sets, and 10 of the remaining sequences were then placed into the test sets because they were incomplete.

| Data Set | Number of Sequences in Set Training | Test | Total | Range of Lengths (number of bases) |
|---|---|---|---|---|
| U1 | 25 | 24 | 49 | 121–170 |
| U5 | 16 | 15 | 31 | 78–138 |
| U6 | 22 | 22 | 44 | 53–116 |
| Totals | 63 | 61 | 124 | 53–170 |

Figure 3.3: The raw snRNA sequences [GP88, GRM93] were organized into training and test sets. Each group was approximately halved into two subsets; one half was used to train and the other half to test the group's grammar.

It should be noted that some sequences may contain structure not present in the referent yeast sequence, which would be undocumented in these trusted multiple alignments. In general it is difficult to know *a priori* which aspects of the secondary structure are important (for example, which base pairs or bulges or individual nucleotides are more important than others in a long helix, such as Stem II of U1).

To produce raw data sets, the aligned sequences were stripped of the inserted characters that aligned them with respect to each other, leaving strings composed only of the four nucleotides {A, G, C, U}. Before the raw sequences were separated into training and test sets, 7 extra-long sequences were omitted (see Figure 3.2) because locally available computers lacked the core memory to parse sequences more than about 175 bases long using these rather complex grammars, which have about 2000 productions and about 500 nonterminals.[1] Then 6 sequences containing N characters were removed from the sets, as N indicates a nucleotide whose exact nature is unknown. Next, 10 incomplete sequences (fragments or substrings of full sequences, when the full sequence is unknown) were placed in the test sets for each set. Although their presence in a training set likely would degrade the quality of the corresponding trained grammar, these incomplete sequences were kept in the test set so the corresponding trained grammar's ability to discriminate and fold them could be assessed. The remaining 114 snRNA sequences, ranging 96–170 bases in length, were divided at random with equal probability into training and test sets (adjusting for

[1] The tRNA grammars developed in previous SCFG work at UC Santa Cruz [SBU+93, SBM+94, SBH+93] had only 1028 productions and 256 nonterminals and modeled sequences only 71–90 bases long.

the number of incomplete sequences already present in each test set) for each grammar (Figure 3.3). This thesis refers to the snRNA sequences by abbreviations of their Latin names, such as `saccer` for *Saccharomyces cerevisiae*.

**2.** A total of 1450 non-snRNA sequences were generated from the National Center for Biotechnology Information's GenBank database (version 75.0+, dated 18 February 1993). This was done by cutting non-snRNA features—including CDS, tRNA, rRNA, mRNA, LTR, D-loop, introns, exons, transposons, miscellaneous features and repeat regions—into snRNA-sized lengths. Specifically, 10 non-snRNA sequences were created for each sequence length between 30 and 174 bases, resulting in a set of 1450 non-snRNA sequences. Any unusual characters (i.e., not `A`, `G`, `C`, `T`, or `U`) appearing in the non-snRNA features were skipped. (Of the total 17,079 RNA lines in the source GenBank file, 81 contained `N` characters and 1 contained a single `Y` character.) Any thymine base (`T`) appearing in a DNA feature was translated to the RNA equivalent, uracil (`U`), in the resulting non-snRNA sequence.

For each grammar, this basic non-snRNA sequence set was augmented with 10 other sequences to form an *augmented non-snRNA set*. The other sequences consist of five complete sequences chosen at random with equal probability from each of the *unmodeled* sets. For example, for the `U5` grammar, the augmented non-snRNA set consists of the 1450 feature fragments plus five sequences from each of the `U1` and `U6` data sets. These complete sequences were included to enrich the set of fragment sequences to better demonstrate the grammars' ability to discriminate false from true examples.

Providing a grammar with a larger set of non-snRNA sequences might enable it to discriminate them better from the snRNA set it was trained to model, as the discrimination criterion relies on obtaining reliable average-NLL scores for a large range of sequence lengths (large enough that the lengths of the modeled set lie within the range). However, larger sets of non-snRNA sequences require more parsing computation time. As a compromise, the set of 1450 non-snRNAs was used. (Section 3.3 includes plots for average-NLL scores for each augmented non-snRNA set as computed by each trained grammar.)

**3.** A total of 32 new unfolded snRNA sequences were retrieved from the latest updates to the aforementioned GenBank database: versions 82.0+, dated 26 May 1994, and 82.0, dated 8 April 1994. This set contains unique sequences—none appear in the training or test sets for any of the grammars—for which only the primary structures are known. The trained grammars were used to determine the base pairing that defines the secondary structure of these snRNA sequences and to discriminate these sequences from an augmented non-snRNA set. These sequences are referred to as the *new* sequences.

## 3.2 Multiple alignments and secondary structure

From a grammar it is possible to obtain a multiple alignment of all sequences. The grammar can produce the most likely parse tree for each of the sequences to be aligned, yielding an alignment of all the nucleotides that align to the match nonterminals in the grammar. Between match nonterminals there can be insertions of varying lengths, but by inserting enough spaces in all the sequences to accommodate the longest insertion, an alignment can be obtained. Because only the number of loop nucleotides varies, while the helix lengths (maximum number of helix-match nonterminals) are fixed, producing a multiple alignment involves merely padding particular sequences with space characters

| **Initial** | Trunk | Stem I | Stem II | Stem III | Stem IV |
|---|---|---|---|---|---|
| Both sides | 42 ( 86%) | 37 (76%) | 3 ( 6%) | 46 ( 94%) | 15 (31%) |
| One side | 45 ( 92%) | 44 (90%) | 21 ( 43%) | 48 ( 98%) | 37 (76%) |
| Some overlap | 49 (100%) | 47 (96%) | 49 (100%) | 49 (100%) | 46 (94%) |

| **Trained** | Trunk | Stem I | Stem II | Stem III | Stem IV |
|---|---|---|---|---|---|
| Both sides | 44 (90%) | 45 (92%) | 0 ( 0%) | 44 ( 90%) | 11 (22%) |
| One side | 47 (96%) | 48 (98%) | 20 ( 41%) | 49 (100%) | 28 (57%) |
| Some overlap | 48 (98%) | 48 (98%) | 49 (100%) | 49 (100%) | 46 (94%) |

Figure 3.4: Each column corresponds to one of the five `U1` helices. The first row shows the number of sequences of the total 49, training plus test, for which the `U1` grammar predicted exactly the same helix boundaries as the trusted alignment; the second, the number of sequences for which the grammar predicted exactly the same boundaries for at least one side of a helix; and the third, the number of sequences for which there is some overlap between predicted and trusted alignments for that helix side, although the boundaries for both sides may differ. For comparison, the bottom table lists the same quantities for the initial grammar, trained on zero sequences.

such that nucleotides produced by the same helix-match nonterminals appear in the same column.

For each trained grammar, a multiple alignment was produced for its corresponding test and training sets. As the following subsections describe in detail, the trusted alignments agree substantially with the trained grammars' predicted alignments. Boundaries for the helices and loops are similar between the predicted and trusted alignments for each set, even for the previously unseen test sequences. (Appendix A shows the three multiple alignments in their entirety.) Figures 3.5, 3.6, 3.10 and 3.14 show in XRNA format predicted foldings for some representative sequences.

### 3.2.1 `U1` predicted alignment

The table in Figure 3.4 tabulates, for each helix in the full `U1` data set, the number of sequences for which there is some agreement between the trained grammar's predicted alignment and the trusted alignment for that helix. See Section A.1 for the complete `U1` multiple alignment. From top to bottom, the rows in the table correspond to decreasing strictness of agreement between the two alignments. A separate table lists the same quantities for the initial grammar for comparison.

Disagreements are focused locally in Stems I, II and IV rather than dispersed globally. Because bulges were modeled between two fixed helix-match positions in the grammar (see Section 2.5), if the trusted alignment places a bulge in Stem I or Stem IV in an uncommon location, often the grammar is unable to capture that shift in bulge position. Also, to make the grammar smaller and easier to train, the Stem II bulge was not incorporated at all in the initial grammar, though the trusted alignment has a bulge of length at least one base for that stem for many sequences (28 of the 49 sequences). Better modeling of bulges might yield alignments that more closely match the trusted alignments.

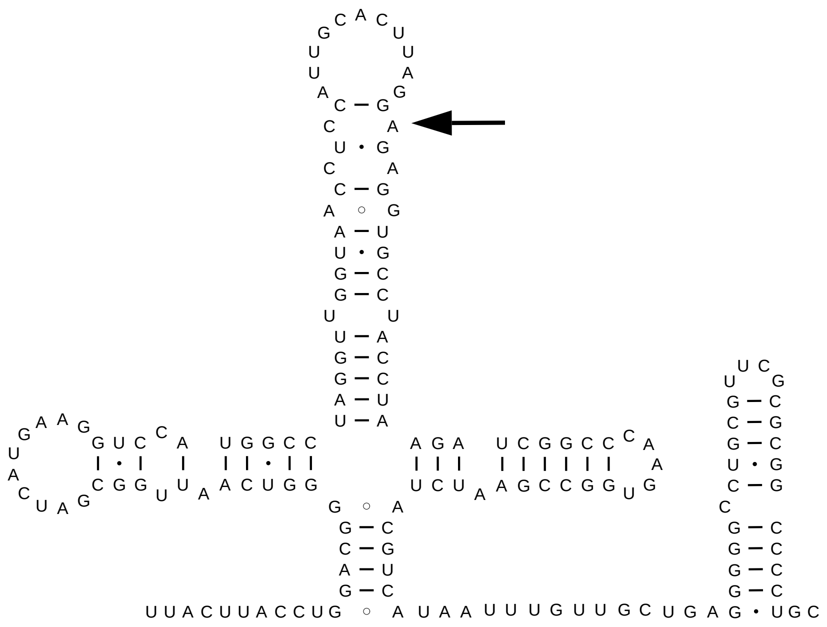


Figure 3.5: The trained `U1` grammar predicted this folding for the sequence `lycesc14`. It differs from the trusted folding only in Stem II, because the grammar did not model the bulge that the trusted alignment placed on the right side; in the trusted alignment, the arrowed `A` comprises the bulge, such that the two `G` nucleotides above it form Watson–Crick base pairs with the `C` nucleotides on the top left.

The trained grammar's predicted alignment suggests some plausible alternative base pairing schemes in Stem IV. Though 38 predicted alignments had Stem IV boundaries different from those in the trusted alignments, 24 of them contained more Watson–Crick base pairs than in the trusted. Figure 3.7 shows three sample sequences for which the predicted alignments contain more Watson–Crick base pairs than their trusted alignments.

Though the predicted sequence alignments differ slightly from the trusted, this trained grammar can discriminate every one of the `U1` sequences, both training and test sequences, from the augmented non-snRNA sequences (even with the limitation that the Stem II bulge was not modeled). In contrast, the initial grammar misclassifies 36 of the 49 sequences in discrimination experiments.

The initial `U1` grammar predicts a similar alignment that in general matches about the same number of helix boundaries with the trusted alignment that the trained `U1` grammar matches. In the trained grammar's predicted alignment, fewer exact helix boundary matches occur but these matches are distributed across slightly more sequences, whereas in the initial grammar's predicted alignment, particular sequences are more closely aligned with the trusted but others do not match well at all. For example, the initial grammar succeeds in aligning both sides of Stem II for three sequences (`triaes1.5*`, `lytvar1G`, and `lytvar2G`), whereas the trained grammar fails to align both sides of Stem II exactly the same as in the trusted alignment for any sequences, but overall the trained grammar's predicted alignment matches the trusted alignment exactly for a total of 144 helices over all the `U1` sequences,

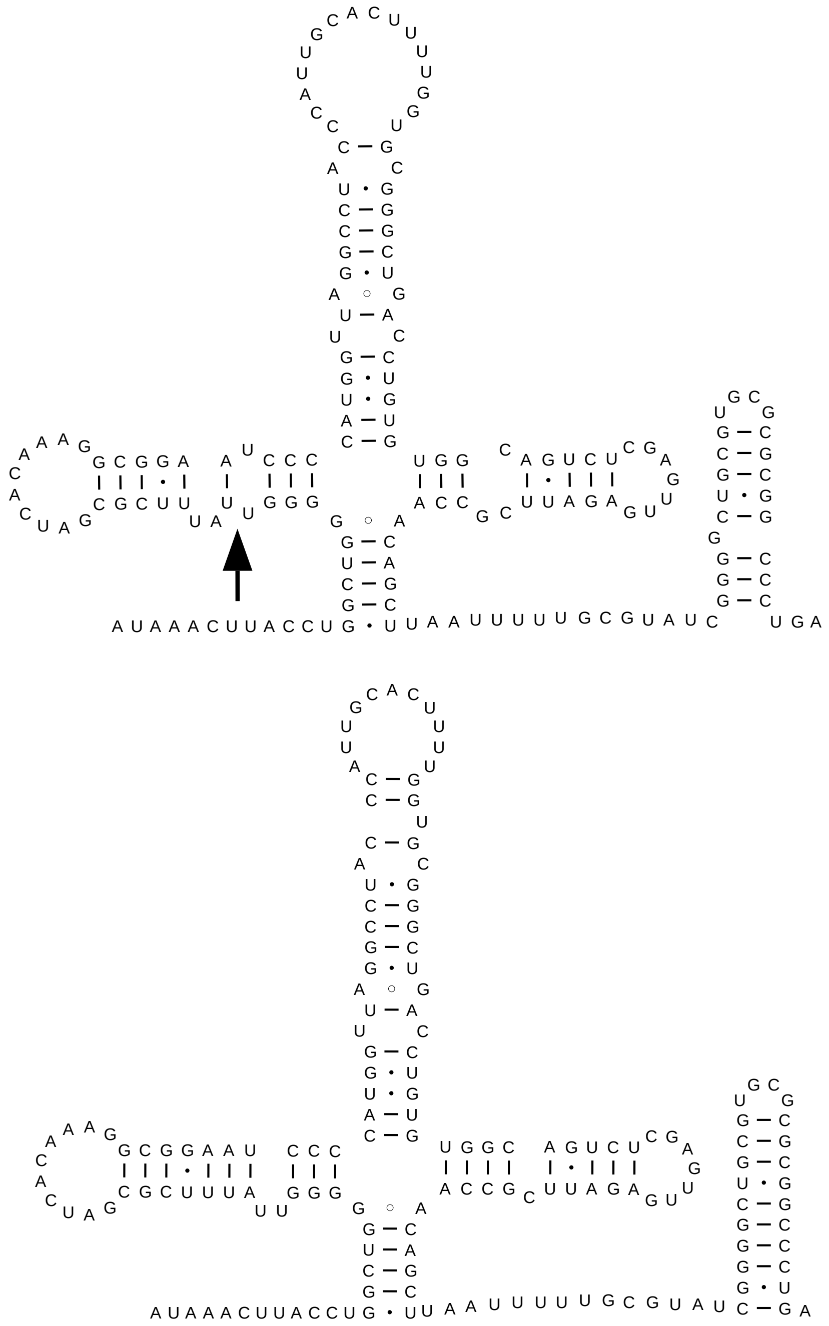


Figure 3.6: The trained `U1` grammar predicted the top folding for the sequence `caeele1`, which is similar to the trusted folding of the same sequence (bottom). Because bulges were modeled in Stem I only between the fifth and sixth helix-match positions, the grammar does not capture the trusted alignment's uncommon positioning of the Stem I bulge between the helix's third and fourth bases (the arrowed `U` nucleotides comprise the trusted alignment bulge).

```
                     Stem II                  Stem II              Stem IV            Stem IV
                 ((((((((((((((((        ))))))))))))))))       (((((   ((((((      )))))))))))
lycesc15  ...GCC.UAGGUUGGUGACUUUC.A...UG.GAGGGGUGCCCGCCUA....GA.GGGGG.C.-AUGCG.UUCG.CGCAG-CCCCU.GC-
                 <<<<<<<<<<<<<<<<      > > >>>>>>>>>>>>>>       <<<<< <   <<<<      >>>>> >>>>>
pissat1C  ...GGC.UAGGCAAGUGACUUCC.A...AG.GAGGGGUGCUUGCCUA....GU.GGGGG.C.-CUGCG.UUCG.CGCGG-CCCUC.UU-
                 <<<<<<<<<<<<<<<<      > > >>>>>>>>>>>>>>       <<<<< <  <<<<<      >>>>> >>>>> >
phavul    ...GCC.UAGGGAAGUGACCUUC.A...AG.AAGGGGUGCUACUCUA....GU.GGGGG.C.-CUGCG.UUCG.CGCGG-CCCCU.UA-
                 <<<<<<<<<<<<<<<<      > > >>>>>>>>>>>>>>       <<<<< <  <<<<<      >>>>> >>>>> >
```

Figure 3.7: Only the differing sections of the predicted and trusted `U1` alignments are shown for these three `U1` sample sequences. Periods indicate elisions, while hyphens mark insertions. The predicted alignment is indicated with parentheses over all basepairing columns, while the trusted alignment for each sequence is indicated by triangle braces below its basepairing positions. The predicted alignment places more Watson–Crick base pairs in Stem II for `lycesc15` and in Stem IV for `pissat1C` and `phavul` than does the trusted alignment.

while the initial grammar has exact matches in 143 helices over all sequences.

### 3.2.2 `U5` predicted alignment

Disagreements between the predicted and trusted `U5` alignments are not globally dispersed, but rather focused in Stems Ia and II. (See the top table in Figure 3.8.) For sequences having more than four nucleotides aligned in Stem Ia in the trusted alignment, the predicted alignment generally (for 10 of the 12 such sequences) managed to align some of the nucleotides within at least one side of the helix to match the trusted alignment. Thus, although the grammar has a limitation in that the number of helix-match nonterminals is fixed, and the grammar was designed with the built-in restriction that no more than four match nonterminals are allowed to model Stem Ia (see Section 2.5), the grammar managed to place some of the nucleotides in the generally correct location. The same can be said for the sequences having more than seven nucleotides aligned in Stem II in the trusted alignment: The predicted alignment generally (for 19 of the 20 such sequences) managed to align some of the nucleotides within at least one side of the helix to match the trusted alignment. Although the `U5` grammar models bulges only between fixed helix-match positions, it did manage to align some of the nucleotides within at least one side of helices containing bulges to match the trusted alignment (for all 7 of the sequences containing bulges). Thus, it seems likely that a grammar tailored to more "perfectly fit" this set (see Section 2.5) might be trained to more closely match the trusted multiple alignment. See Section A.2 for the complete `U5` multiple alignment.

The `U5` predicted alignment provides several plausible alternative structures. For two sequences, `crycoh` and `tetthe51`, the `U5` grammar's predicted alignment contains more Watson–Crick base pairs in Stem II than the trusted alignment does. For `crycoh`, the predicted alignment places some of a `U`-rich strand in the helix instead of in the adjacent loop. The two foldings for this sequence are depicted side by side in graphical XRNA form in Figure 3.9.

For `tetthe51`, the predicted alignment has an additional `A-U` pair, making the helix as long as the most common length. For the other sequences in which the Stem II predicted alignment differed from the trusted, in two cases the predicted adds an `A-C` pair, in one it adds a `U-G` pair, and in the others it either adds `U-U` or `G-G` which are not base pairs, or, if the trusted alignment of Stem II contains more than seven base pairs, slides any

| **Initial** | Stem Ia | Stem Ib | Stem Ic | Stem II |
|---|---|---|---|---|
| Both sides | 19 (61%) | 22 ( 71%) | 24 (77%) | 7 ( 23%) |
| One side | 19 (61%) | 23 ( 74%) | 25 (81%) | 7 ( 23%) |
| Some overlap | 30 (97%) | 31 (100%) | 30 (97%) | 31 (100%) |

| **Trained** | Stem Ia | Stem Ib | Stem Ic | Stem II |
|---|---|---|---|---|
| Both sides | 19 (61%) | 28 (90%) | 24 (77%) | 8 (26%) |
| One side | 19 (61%) | 29 (94%) | 30 (97%) | 8 (26%) |
| Some overlap | 30 (97%) | 30 (97%) | 30 (97%) | 30 (97%) |

Figure 3.8: Each column corresponds to one of the four U5 helices. The first row shows the number of sequences of the total 31, training plus test, for which the U5 grammar predicted exactly the same helix boundaries as the trusted alignment; the second, the number of sequences for which the grammar predicted exactly the same boundaries for at least one side of a helix; and the third, the number of sequences for which there is some overlap between predicted and trusted alignments for that helix side, although the boundaries for both sides may differ. For comparison, the top table lists the same quantities for the initial grammar, trained on zero sequences.

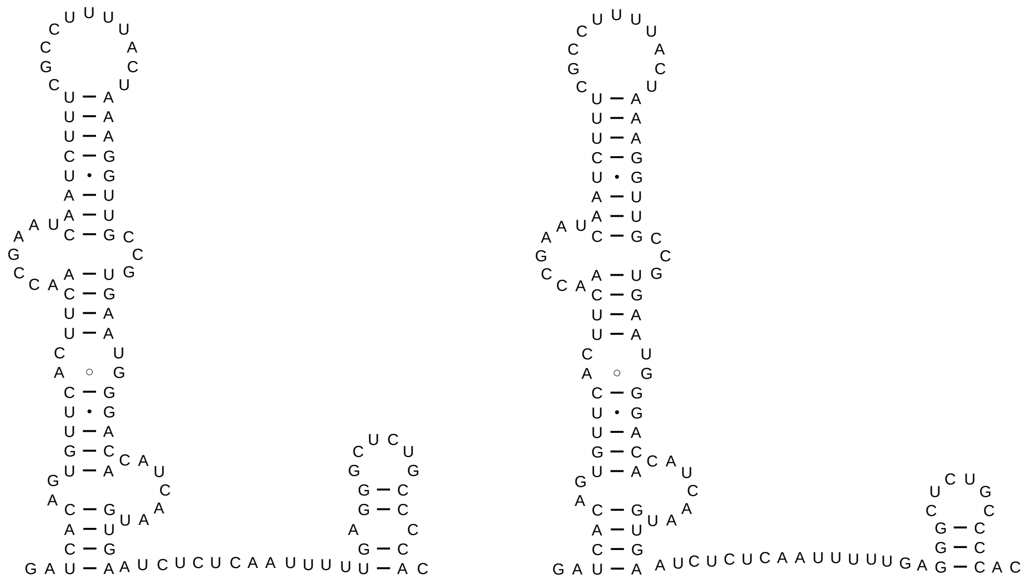


Figure 3.9: In Stem II for the crycoh sequence, the U5 grammar's predicted folding (left) contains four Watson–Crick base pairs whereas the trusted folding (right) contains only three.

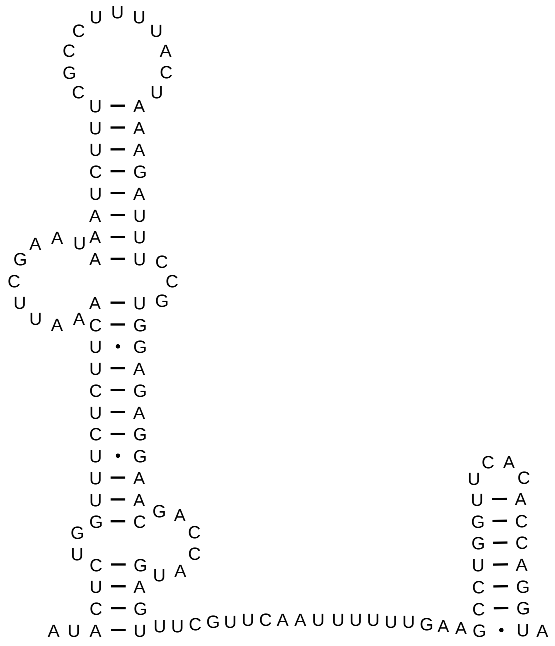


Figure 3.10: This folding was predicted by the trained `U5` grammar for the `xenlae` sequence. It matches the trusted folding of this sequence exactly.

```
                   Ia     Stem Ib             Stem Ib         Ia           II              II
                  ((((  (((((((((((         )))))))))))      ))))         ((((            ))))
schpom    ...UCCGUCAAAG-CACUUUGCAAAAGC...CCGUUUGUAAGGUGUGCUAAUUUGACU...GAAUCUU----UUUC----UUGAAA
                <<<<<<  <<<<<<<<<<          >>>>>>>>>>>      >>>>>>         <<    <<        >>>>
aratha    ...ACGCAGCCAUGUGGUGAGUACAAAG...CCGUGUGCUCUCGACGCUAAGUGCAUA...GAGGGCUCCAC...UGUGGAACCCAA
             <<<<<      <<<<<<<<<<<         >>>>>>>>>>        >>>>>      <<<<<<<<<    >>>>>>>>>
homsap5d*       AUACUCUGGUUUCUCUUCAAAU...CCGUGGAGAGAAACCGUUUUGAGUUUC...UGA--AG-----------CCC--U
                  <<<<  <<<<<<<<<<<         >>>>>>>>>>>      >>>>            <           <<<  <
```

Figure 3.11: Only the differing sections of the predicted and trusted `U5` alignments are shown for these three `U5` sample sequences. Periods indicate elisions, while hyphens mark insertions. The predicted alignment is indicated with parentheses over all basepairing columns, while the trusted alignment for each sequence is indicated by by triangle braces below its basepairing positions. (The trusted alignment places the last five bases of `homsap5d*` all on the left side of Stem II.)

"extra" bases into the adjacent Loop II. Sample predicted and trusted alignments for three `U5` sequences are shown in Figure 3.11. For each of the two sequences for which the Stem Ib alignments differ (`schpom` and `aratha`), the predicted alignment contains one more base pair than the trusted. Finally, although the predicted alignment bounds Stem Ia differently for 12 sequences, its Stem Ia alignment has the same number of Watson–Crick base pairs as the trusted for 26 of the total 31 sequences, among them eight of those 12 sequences.

The initial grammar, trained with only Dirichlet pseudocounts, predicts a fairly similar alignment. (See the bottom table in Figure 3.8.) Interestingly, for the incomplete sequence `caimos*`, the initial grammar manages to align some of the same bases in Stems Ib and II that appear in the trusted alignment, whereas the trained grammar matches none of the trusted alignment's helix positions for `caimos*`. (The trained grammar misclassifies

| | **Initial** | **Trained** |
|---|---|---|
| Both sides | 0 ( 0%) | 0 ( 0%) |
| One side | 0 ( 0%) | 0 ( 0%) |
| Some overlap | 34 (77%) | 39 (87%) |

Figure 3.12: The first column corresponds to the initial grammar (for comparison), and the second to the trained U6 grammar. The first row shows the number of U6 sequences of the total 44, training plus test, for which the grammar predicted exactly the same helix boundaries as the trusted alignment; the second, the number of sequences for which the grammar predicted exactly the same boundaries for at least one side of the helix; and the third, the number of sequences for which there is some overlap between predicted and trusted alignments for that helix side, although the boundaries for both sides may differ.

this sequence, as well, in discrimination experiments.) However, for 6 sequences, the initial grammar places the boundaries of Stems Ib and Ic one match position off (with respect to the trusted alignment), whereas the trained grammar's predicted alignment matches the trusted exactly for those sequences. The number of base pairs is equal in either case, though. Also, for the incomplete sequence homsap5d* the trained grammar manages to place some of the same bases on one side of Stem II that the trusted places there, whereas the initial grammar fails to place any of those same bases in the helix.

Of the two incomplete sequences included in the test set, only the dramatically shorter of the two, caimos*, was completely misaligned by the trained grammar. The other, homsap5d*, was aligned correctly for all but Stem II—indeed, it was deemed an incomplete sequence because it lacks both Loop II and one side of Stem II in the trusted alignment.

### 3.2.3 U6 predicted alignment

The U6 predicted alignment places boundaries for the single U6 helix and loop in different positions, compared to the trusted alignment, for every one of the 44 U6 sequences, training plus test. However, the grammar manages to align some of the nucleotides within at least one side of the helix to match the trusted alignment for many sequences (39 of the 44). (See the table in Figure 3.12.)

Further, for all sequences, the grammar does locate a stem-loop structure near the $5'$ end, reflecting the fact that some features in the data encourage the base-pairing productions to become more likely than the skip productions during re-estimation of the production probabilities. As mentioned in Section 3.1, more structure may be present in the sequences than is captured in the trusted alignments because the sequences were not examined for individual motifs. See Section A.3 for the complete U6 multiple alignment.

Several of the sequences (34 of the 44) have bulges of varying lengths in the helix in the trusted alignment, but for computational ease of training the grammar was not designed to model bulges at all. Also the helix length chosen for the U6 grammar was limited to eight match positions (see Section 2.5), whereas the trusted alignment places more than eight nucleotides on one or both sides of some sequences (13 of the 44). The limited similarity of the U6 predicted alignment to the trusted alignment stems from these restrictions on the initial grammar. (See Figure 3.13 for the predicted and trusted alignments of three representative sequences.)

```
             ((((((((      ))))))))
saccer      CUUCCCGGAUUAACGUCCGUGGAACA...
            <<<<<<<<<<    >>>>>>>>>>
zeamay    GUCUCUUCGGA-GACA-UCCGAUAAAAU...
          <<<<<    >> >>>
vicfab      C--UUCGGG-GACA-UCCGAU--AAA...
            <      >> >>>
```

Figure 3.13: Only the differing sections of the predicted and trusted alignments are shown for these three sample sequences. Periods indicate elisions, while hyphens mark insertions. The predicted alignment is indicated with parentheses over all basepairing columns, while the trusted alignment for each sequence is indicated by by triangle braces below its basepairing positions. In the two bottom sequences, the predicted alignment contains an equal or greater number of Watson–Crick base pairs relative to the trusted alignment.

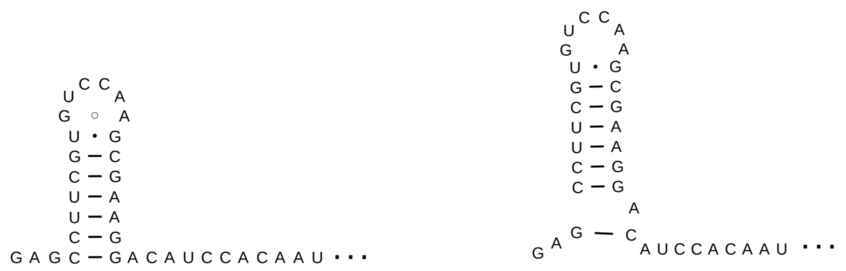


Figure 3.14: The left folding was predicted by the trained `U6` grammar for the incomplete `trycru*` sequence's helix. In contrast, the trusted folding (right) adds a bulge at the 24th nucleotide `A` to fit in an additional `G-C` pair.

For the incomplete sequence `trycru*`, the trained grammar did predict a folding that is fairly similar to its trusted folding. (See Figure 3.14.) For the other incomplete sequence `leicol*`, however, the predicted and trusted foldings differ substantially (the similar-length helix was shifted by more than three positions).

The `U6` predicted alignment also provides several plausible alternative structures. For 14 of the 44 sequences, the predicted alignment makes the helix contain as many or more Watson–Crick base pairs than are present in the trusted alignment for those same sequences, even when the trusted alignment includes bulges. Despite the difference between predicted and trusted multiple alignments for the `U6` sequence set, the trained grammar is able to discriminate all complete `U6` sequences from non-snRNA sequences of similar length, whereas the initial grammar is unable to discriminate any of them. (See Section 3.3.) For example, Figure 3.15 shows in XRNA format the predicted and trusted alignments for the `saccap` sequence. This suggests that, in discrimination of `U6` sequences, representation of a stem-loop structure may be more important than its exact nature or precise location.

The initial `U6` grammar's predicted alignment is similar to the trained grammar's alignment. Even with no training, it folds 13 sequences to contain more Watson–Crick base pairs than in the trusted alignment of those sequences. The initial grammar's alignment of `trybru` more closely resembles the trusted alignment than the trained grammar's align-

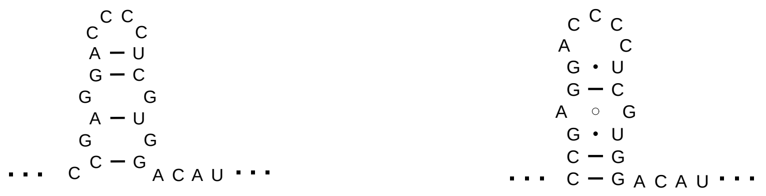


Figure 3.15: The predicted folding (left) for the U6 helix for saccap contains more Watson–Crick base pairs than the trusted folding (right) for that same portion of the same sequence, although the trusted folding contains G-U and A-G pairs.

ment, but in general the trained grammar's alignment more closely matches the trusted (see Figure 3.12). However, it is unable to discriminate any of the 44 sequences from non-snRNA sequences.

Training on the 22 training sequences allowed the grammar to align sequences with more Watson–Crick base pairs than before it was trained. The trained grammar's predicted alignments for 13 sequences contain more Watson–Crick base pairs than the initial grammar's predicted alignments for them, while the initial grammar's predicted alignments for only 7 sequences contain more Watson–Crick base pairs than the trained grammar's alignments for them.

### 3.2.4 Folding new sequences

All trained grammars were used to predict foldings for their test sets (see previous subsections), but a trained grammar can also predict foldings for sequences for which only the primary sequences are known. As an additional test, the grammars were used to fold 32 sequences drawn from the latest GenBank updates (82.0+, dated 26 May 1994, and 82.0, dated 8 April 1994) for which only primary sequences are known. These are referred to as *new* sequences.

Figures 3.16, 3.17 and 3.18 show multiple alignments for the 12 new U1 sequences, 6 new U5 sequences and 14 new U6 sequences, respectively, as predicted by the three corresponding trained grammars. Parentheses over the base-pairing columns locate the helices. Periods denote ends of helices and sometimes insertions. Hyphens represent insertions. Lowercase letters in the U5 and U6 alignments indicate applications of insert productions; the loops are not aligned in the U1 alignment.

The U1 sequence ratnor seems oddly to have been aligned too far to the left relative to the other new sequences' predicted alignments, such that it appears to have an unusually long lead loop on its $5'$ end and it appears to end prematurely with no Stem IV or Loop IV. This may be an artifact from its designation in the GenBank database. It is not discriminated either.

The U5 sequences caeele2 and musmus, both appear to have slightly unusual alignments, with gaps inserted in some helices. These two may be incomplete sequences (fragments), they may just have unusual structure, or they may have been mislabeled in the GenBank database.

The three mycoplas sequences listed in the U6 alignment are not actual U6 sequences, but rather sequences that biologists have flagged as resembling U6. The trained U6 grammar aligns portions of those sequences to the helix such that their alignments closely resemble

```
                                                    Trunk  Stem I w/Bulge              Stem I
                                                    (((((( (((((    (((((            ))))))))))
asclum1  --------------------------------AAACUAACCU.GGCUGG.GGGGC.AU.CUCGC.GAUCAUGAAG.GCGGGACCUC.
asclum2  --------------------------------AGACUUAUUU.GGUUGG.GAGGA.U-.UUCGU.AAUCAGAAG-.GCGGGACCUC.
asclum3  --------------------------------AAACUUACCU.GGCUGG.GAGGC.U-.UUCGU.GAUCAUGAAG.GCGGAACCUC.
caeele   --------------------------------AAACUUACCU.GGCUGG.GGGUU.AU.UUCGC.GAUCACAAAG.GCGGAAUCCC.
lycesc1  --------------------------------AUACUUACCU.GGACGG.GGUCA.A-.UGGGC.GAUCAUGAAC.ACCCAUGGCC.
lycesc2  ---------------------------------UACUUACCU.GGACGG.GGUCA.A-.UGGGC.GAUCAUUAAG.ACCCAUGGCC.
musmus   --------------------------------AUACUUACCU.GGCAGG.GGAGA.U-.ACCAU.GAUCACGAAG.GUGGUUUUCC.
ratnor   CAGGGGAGAGCGCGAACGCAGUCCCCCACUACCACAAAUUAU.GCAGUC.GAGU-.U-.-UCCC.GCAUUUG---.GGGA-AAUCG.
schman1  --------------------------------AAACUUACCU.GGCGCC.GGGUU.C-.AGGGU.GAUCAGGAAU.GCUCUGACCC.
schman2  --------------------------------AAACUUACCU.GGCGCC.GGGUU.C-.AGGGU.GAUCAGGAAU.GCUCUGACCC.
tetthe1  -----------------------------------AUUACAA.UGUUGU.AGUUA.G-.CUAUA.UAUCAAAAAA.UAUAGCAACU.
tetthe2  ----------------------------------ACUUACCU.GGCUGG.AGUUA.G-.CUAUC.GAUCAUGAAG.GGUAGCGGCU.

              Stem II                         Stem II     Stem III               Stem IIIw/Blg
         ((((((((((((((((                ))))))))))))))))) (((((((((              ))))))    )))
asclum1  CAUGGUGAGGUCUGGU.CAUUGCACUUCCG-.ACCAGGCUGACCUGUG.UGGCAGUCC.CGAGUUG-----.GGAUUG.G-.CCA.
asclum2  CAUGGCGAGGCUUGGU.CAUUGCACUUUCG-.ACCAGGCUGACCAGUG.UGGCAGACC.CGAGUUG-----.GGAUUG.G-.CCA.
asclum3  CAUGGUGAGGCUUGGU.CAUUGCACUUUCG-.ACCAGGCUGACCCGUG.UGGCAGUCC.CGAGUUG-----.GGAUUG.G-.CCA.
caeele   CAUGGUUAGGCCUACC.CAUUGCACUUUUGG.UGCGGGCUGACCUGUG.UGGCAGUCU.CGAGUUG-----.AGAUUC.G-.CCA.
lycesc1  UAGGUUGGUGACCAUC.AUUGCACUUUG---.AAGGGGUGCCCGCCUA.AGGUCGGCC.CAAGU-------.GGUCGA.G-.CCU.
lycesc2  UAGGCUUGUGACCUCC.AUUGCACUUUG---.GAGGGGUGCCUGCCUA.AGGUUGGCC.CAAGU-------.GGUCGA.G-.CCU.
musmus   CAGGGCGAGGUGUAUC.CAUUGCAUCCG---.GAUGUGCUGACCACUG.CGAUUUCCC.CAAAUGC-----.GGGAAA.C-.UCG.
ratnor   CAGGGGUCAGCACAUC.CGGAGUGCAAUG--.GAUAAGCCUCGCCCUG.GGAAAACCA.CCUUCGUGAUCA.UGGUAU.C-.UCC.
schman1  CGGGUGGAGGCUCAC-.CAUUGCACUUCG--.-GUGGGUUGAAACUUG.CGACGAACC.CUAAUUG-----.GGUGCG.C-.UCG.
schman2  CGGGUGGAGGCUCAC-.CAUUGCACUUCG--.-GUGGGUUGAAACUCG.CGACGAACC.CUAAUUG-----.GGCUCG.C-.UCG.
tetthe1  AAGGUGGAGCAAGUC-.AUUGUACUAAAGA-.-UGUUUGUAAUACCUU.GAUGUUCCC.GCU---------.GGGAGC.A-.AUA.
tetthe2  UAGGGUGGAGCAGGUC.AUUGCACAAAAGA-.UGUCUGUAAUACCUUA.UUGUUCCCC.GUGC--------.GGGGAA.CC.GAA.

          Trunk                       Stem IVw/Bulge        Stem IV
         ))))))                       (((((   ((((((      )))))))))))
asclum1  AGAGCA.UAAUUUUUGCGU---------.UUGGG.G.ACAGCG.UUCG.CGCUUCCCCGC.CC
asclum2  ACAGCA.UAAUUUUUGCGU---------.UUGGG.G.ACAGCG.UUCG.CGCUUCCCCGC.GU
asclum3  ACAGCA.UAAUUUUUGCGU---------.UUGGG.G.ACAGCG.UUCG.CGCUUCCCCGC.CC
caeele   ACAGCU.UAAUUUUUGCGU---------.AUCGG.G.GCUGCG.UGCG.CGCGGCCCUGA.A
lycesc1  ACGUCA.UAAUUUGUUGCUGA-------.GGGGG.C.CUGCG-.UUCG.-CGCGGCCCCU.GC
lycesc2  ACGUCA.UAAUUUGUUGCUG--------.UGGGG.G.CCUGCG.UACG.CGCAGCCCCUG.CC
musmus   ACUGCA.UAAUUUGUGGUAGU-------.GGGG-.G.ACUGCG.UUUG.UGCUCU-CCCC.UU
ratnor   -CCUGC.CAGGUAAGUAU
schman1  GACGCG.UAGUUUUUGUCAG--------.UGG--.G.GAGGUC.UUCG.--GAUCAUCCC.UU
schman2  GACGCG.UAGUUUUUGUCAG--------.UGG--.G.GAGGUC.UUCG.--GAUCAUCCC.UU
tetthe1  ACAACA.AAAUUUCUGAUUGGAAAUAGU.CAUUA.A.ACUAAC.UG--.GCUAUUUCCUC.U
tetthe2  ACAGCA.CAAUUUCUGCUAGG-------.GGAGA.C.GUGCAC.UUA-.GUGCUGUCUCC.GCU
```

Figure 3.16: The trained U1 grammar predicted these foldings for 12 new sequences taken from the updated GenBank database. (Loops are not aligned.)

the aligned portions of other actual U6 sequences, such as soltub and stropur. However, it does not discriminate these sequences as being U6 sequences.

XRNA representations for predicted foldings of sample sequences from the set of new sequences are shown in Figures 3.19, 3.20 and 3.21. The U1 figure shows a sequence from the organism *Tetrahymena thermophila*. The U5 figure shows a sequence from each of the two eucaryote worm organisms *Ascaris lumbricoides* and *Caenorhabditis elegans*. The U6 figure shows the helix portions of six sequences including the three pseudo-U6 sequences from the *Mycoplasma* genus.

An empirical examination of the predicted foldings reveals that they do resemble the trusted foldings of other sequences from the same set. Most sequences are also aligned in a manner similar to the sequences appearing in the trusted alignments for the same set. These predicted foldings may be viable secondary structures for the new sequences.

```
                  Ia         Stem Ib                  Stem Ic             Stem Ic        Stem Ib
                 ((((      (((((((((((               ((((((((             ))))))))     )))))))))))
asclum     ...AG.-CUC.U..G.GUUCCUCUGCA.UccAC...CGAGA.AAUUCUUU.CGCCUUUUACU.AAAGAUUU.CCG.UGCAGAGGAAC.
caeele1    ...-A.ACUC.U..G.GUUCCUCUGCA.U..UUaacCGUGA.AAAUCUUU.CGCCUUUUACU.AAAGAUUU.CCG.UGCAAAGGAGC.
caeele2    gaaAA.UCUU.U..U.GCCUUUUACUG.A..AU...-AUUU.--------.----------u.--------.CCU.UGCAAAGGAGC.
musmus     cucUU.ACUG.UccG.GAAUC-CAGUG.G..UC...CUUAG.UGGGCGGG.-----UUUACU.CUGGCAGA.--G.UUUCU-GUUUG.
tetthe1    ...AU.CACA.-..G.AACUCAGCUCA.U..UA...CGCUU.UAAUUUUU.CGCCUUUUACU.AAAGAUUA.CCG.UGGGCUGGGUU.
tetthe2    ...AU.CACA.-..G.AACUCAGCUCA.U..UA...CGAUU.UAAUUUUU.CGCCUUUUACU.AAAGAUUA.CCG.UGAGCUGGGUU.

                      Ia                       Stem II           Stem II
                     ))))                      (((((((           )))))))
asclum     G.UUU.A.U.GAG-.UAUA.CGCCAAUUUUUGGaG.UCCCAGC...UUC...G.GCUAGGG.acaA
caeele1    A.UUUaC.U.GAGU.AUUAcAUACAAUUUUUGG.A.GACUCCU...UGAgaaA.GCGGGUC....A
caeele2    AuACA.U.U.GAGU.AUUAuAUACAAUUUUUGG.A.GUCC-CC...UUG...A.GA-AAGC.g..G
musmus     A.AAU.AgU.UGGU.AUUA.AG----UU-----.G.AUUCUGU.cgCUG...U.UCGGAAU....A
tetthe1    U.ACC.A.A.UGUG.AAUU.AUUAAAAUUUUUG.C.AGGAUUC...UUU...U.GAAUCCU.c..U
tetthe2    U.UUC.A.A.UGUG.AAUU.AUUAAAAUUUUUG.C.AGGAUUC...UUU...U.GAAUCCU.c..U
```

Figure 3.17: The trained U5 grammar predicted these foldings for 6 new sequences taken from the updated GenBank database.

```
                ((((((((       ))))))))
asclum    .....A.UAUAAAUA.U....CUUGUAUA.UUUAUAAUAUUGGC...
homsap    .....U.UUUGUAUC.ACA..UAUACUAA.AAUGGCGCUAGCGA...
lycesc    ...CCG.UACUCGCU.U....CGGCGGUA.CAUAUACUAAAAUU...
mycoplas1 ...GUC.CCUUCGGG.GACA.UCCGAUAA.AAUUGGAACGAUAC...
mycoplas2 .....G.UGUUAGCU.U....CGGCAACA.CAUCUAUUAAAAUU...
mycoplas3 .....A.UAUAAAUA.U....CUUGUAUA.UUUAUAAUAUUGGC...
phypol    .....C.CCGAAAGG.GUCC.UCCGUUAA.AAUUGGAACGAUAC...
schman    AGAGCC.CGAAAGGG.CA...UCUGUUAA.AAUUGGAACGAUAC...
soltub    .GGAGC.CCUUCGGG.GACA.UCCACAAA.CUGGAAAUUCAACA...
stropur   UGGAGC.CCUUCGGG.GACA.UCCACAAA.CUGGAAAUUCAACA...
tetthe*   .....C.CCGAAAGG.GUCC.UCCGUUAA.AAUUGGAACGAUAC...
tetthe    AGAGCC.CGAAAGGG.CA...UCUGUUAA.AAUUGGAACGAUAC...
trybru    .GGAGC.CCUUCGGG.GACA.UCCACAAA.CUGGAAAUUCAACA...
trybru2   UGGAGC.CCUUCGGG.GACA.UCCACAAA.CUGGAAAUUCAACA...
```

Figure 3.18: The trained U6 grammar predicted these foldings for 14 new sequences taken from the updated GenBank database. Only the helix and portions of the adjacent (unaligned) loops are shown.

## 3.3 Discriminating snRNA from non-snRNA sequences

As described in Section 2.2, a NLL score is calculated for each test sequence and is then used to measure how well the sequence fits the corresponding grammar. This raw NLL score depends too much on the test sequence's length to be used directly to decide whether a sequence belongs to the set modeled by the grammar. Thus the raw scores are normalized by calculating the difference between the NLL score of a sequence and the average NLL score of a typical non-snRNA sequence of the same length, measured in standard deviations. This number is called the *Z score* for the sequence [KBM+94]. By choosing a Z-scores cutoff, one can classify sequences with Z scores above the cutoff as being snRNA sequences. While I cannot prove that these normalized scores actually exhibit Gaussian tails for non-snRNAs, this kind of Gaussian approximation has worked well previously [KBM+94, SBU+93, SBM+94, SBH+94b].

To test the ability of the trained grammars to discriminate their snRNA from other RNA sequences of similar length, for each of the trained grammars, calculations were made to obtain the Z score of every sequence in the corresponding snRNA database and every

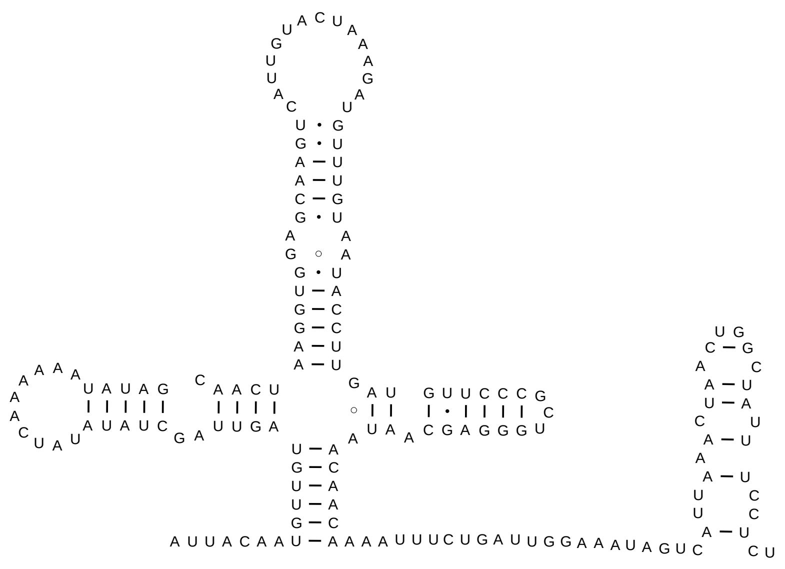


Figure 3.19: The trained `U1` grammar predicted this folding for the sequence `tetthe1`. Its "true" secondary structure is unknown in that no trusted multiple alignment was available. However, it resembles `U1` structures in the trusted alignment.

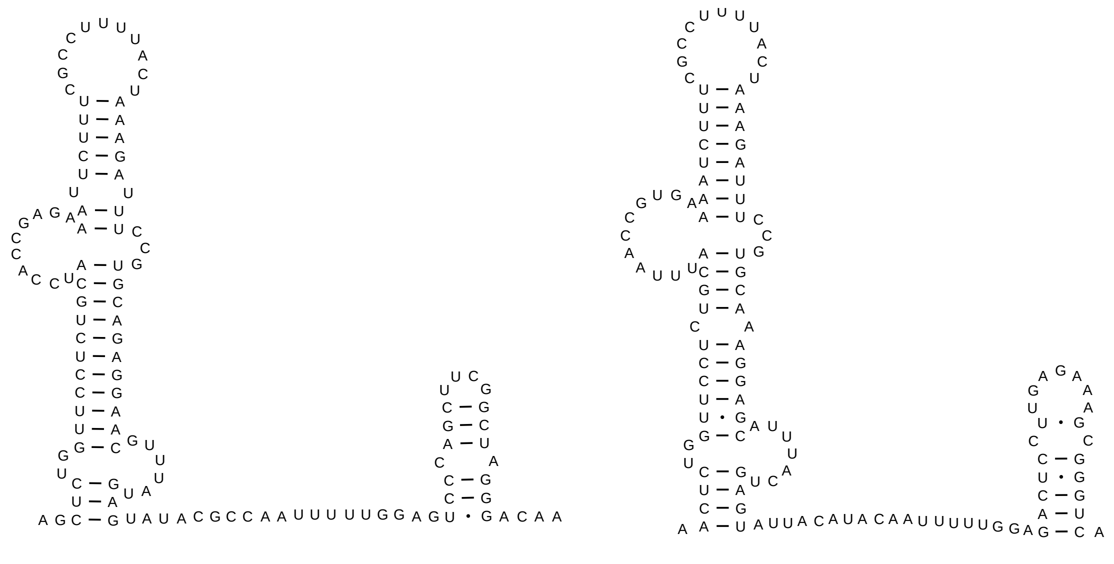


Figure 3.20: The trained `U5` grammar predicted these foldings for the sequences `asclum` (left) and `caeele1` (right). Their "true" secondary structures are unknown in that no trusted multiple alignment was available. However, they resemble `U5` structures in the trusted alignment.

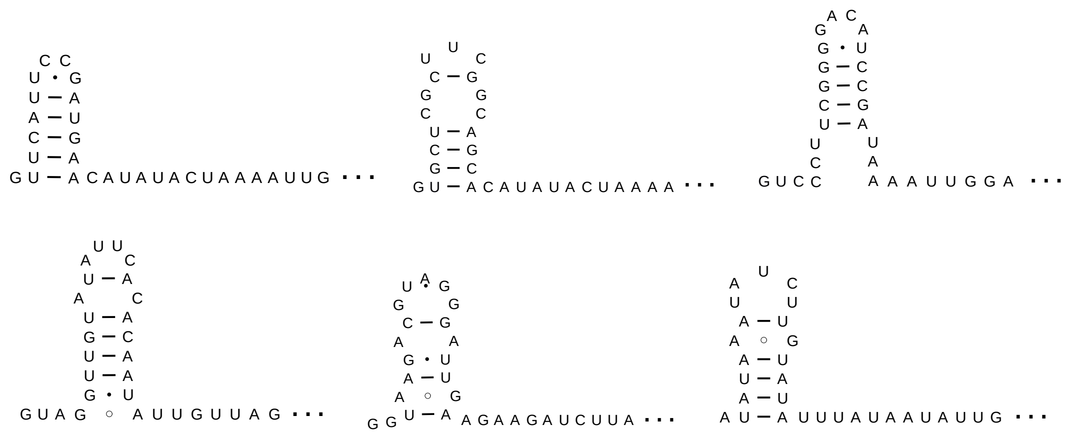


Figure 3.21: The trained U6 grammar predicted these helix foldings for the U6 sequences asclum, homsap and lycesc (top) and the pseudo-U6 sequences mycoplas1, mycoplas2 and mycoplas3 (bottom). Their "true" secondary structures are unknown in that no trusted multiple alignment was available. However, they resemble U6 structures in the trusted alignment.

sequence in an augmented set of non-snRNA sequences. Although the highest Z score of any such non-snRNA is never much greater than 4, an snRNA sequence is not considered to be successfully discriminated from the augmented non-snRNA sequence set unless its Z score is greater than 5. The choice of 5 is motivated by the assumption that the non-snRNAs are drawn from a normal distribution, and for a normal distribution, only about $8 \times 10^{-11}\%$ of the sequences will have Z scores greater than 5. Thus, if the non-snRNA sequences truly adhere to a normal distribution, virtually no non-snRNA sequences—about one in one billion—should have a Z score higher than 5.

Figures 3.24, 3.27, and 3.30 show histograms of the Z scores for the U1, U5 and U6 grammars, respectively, demonstrating the ability of each to discriminate 1460 augmented non-snRNA sequences from sequences it was trained to model. Under each histogram is a plot of the NLL scores for all augmented non-snRNA sequences computed using the trained grammar's parameters, versus their lengths (number of bases). Superimposed on these points is the average-NLL curve used to compute Z scores. Because the average NLL curve was computed using a window of about 40 sequences, an artifact appeared: the average curve appears to diverge from the actual average for sequences shorter than about 30 bases in length, where the window was too small. A perhaps more rigorous tack would be to perform linear regression on the Z scores for the non-snRNA sequences, obtaining an equation to describe the average non-snRNA Z score as a simple linear function of the sequence length. This would give a better approximation of a normal distribution of non-snRNA sequences. Though the divergence does inflate unduly the Z scores for short non-snRNA sequences, it does not skew any snRNA Z scores since no snRNA sequences, either complete or incomplete, are shorter than 53 bases in length.

For each grammar, Figure 3.22 shows the number of snRNA sequences in each set that are successfully discriminated from the augmented non-snRNAs using the criterion that snRNA sequences with Z scores greater than 5 are correctly discriminated. Since none of the 1460 augmented *non*-snRNA sequences had a Z score higher than 5 for any

| | Matching Subunit Set | | | | | | Augmented Non-snRNA Set | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\leq 5$ | $> 5$ | Low Zs | BPB | Name | Null BPB | High Zs | BPB | Name | Null BPB |
| `U1` | 0 | 49 | 5.466 | 2.097 | `schpom` | 2.049 | 3.997 | 2.405 | `U5` | 2.059 |
| | | | 10.944 | 1.645 | `chlsac` | 2.167 | 3.344 | 2.422 | misc. | 2.053 |
| `U5` | 1 | 30 | 3.858 | 2.247 | `caimos*` | 2.081 | 4.034 | 2.167 | misc. | 2.063 |
| | | | 7.454 | 1.854 | `schpom` | 2.058 | 3.349 | 2.269 | tRNA | 2.039 |
| `U6` | 2 | 42 | 2.620 | 3.060 | `leicol*` | 2.108 | 3.482 | 2.472 | repeat | 2.057 |
| | | | 2.843 | 2.790 | `trycru*` | 2.096 | 3.257 | 2.569 | CDS | 2.045 |
| | | | 7.887 | 1.991 | `trybru` | 2.067 | 3.027 | 2.553 | CDS | 2.050 |
| Total | 3 | 121 | | | | | | | | |

Figure 3.22: The table shows how each trained grammar partitions the sequence set it was intended to model based on their Z scores (124 snRNA sequences total). The "$\leq$ 5" column tabulates the number of sequences whose Z scores were no greater than 5 standard deviations, while the "$>$ 5" column shows the number of sequences whose Z scores were greater than 5 standard deviations. The rows show sequences closest to the Z-scores boundary (the highest-scoring non-snRNA and lowest-scoring snRNA sequences). The third and seventh columns show these sequences' Z scores; the fourth and eighth columns show their NLL scores in base 2 divided by sequence length, or *bits per base* (BPB). The fifth and ninth columns label the sequences. The sixth and tenth columns represent the BPB that a *null model* would give for a sequence of that length. The incomplete `*` sequences were neither aligned nor discriminated by the trained grammars.

of the trained grammars, there were no false positives. There was one false positive for the `U6` initial grammar, though. *False positives* are non-snRNAs misclassified as snRNAs, while *false negatives* are snRNAs misclassified as non-snRNAs. The figure shows actual Z scores for sequences closest to the Z-scores cutoff. For generality, it also shows an alternate formulation called *bits per base (BPB)* for those sequences.

The BPB measure is useful for assessing a model's utility from a data-compression point of view. The *bits per base* for a sequence is the sequence's NLL score in logarithm base 2 divided by the sequence's length. (NLL in base 2 is obtained by dividing the raw NLL score, which is in base $e$, by the natural logarithm of 2.) This gives the likelihood in base 2 for an "average" nucleotide in an RNA sequence, which approximates how many bits of computer memory the grammar would require to represent a nucleotide in that sequence. For comparison, Figure 3.22 also shows the BPB scores that a *null grammar* would provide, where the null grammar is defined as the simplest model representing the sequences. The null grammar reflects the information content of the primary sequences, assuming nothing is known about their secondary structure. The null BPB values are calculated by adding to 2 (the number of bits required to specify one of four nucleotides) the logarithm base 2 of the sequence length (the number of bits required to encode the sequence length) divided by the length.

All three trained grammars produce BPB scores much less than 2 for all snRNA sequences except those closest to the Z-scores boundary, indicating that these trained grammars would require less space to represent the sequences they model than would a null grammar. Conversely, the BPB scores for the augmented non-snRNA sets are higher than 2, usually around 3, indicating that the trained grammars would require more memory to represent sequences that they do *not* model than would a null grammar. This result fits

| Grammar | Number of Sequences $\text{BPB} \leq 1.0$ | $1.0 < \text{BPB} < 2.0$ | $\text{BPB} \geq 2.0$ | Total |
|---|---|---|---|---|
| U1 | 35 | 13 | 1 | 49 |
| U5 | 15 | 15 | 1 | 31 |
| U6 | 36 | 6 | 2 | 44 |
| Totals | 86 | 34 | 4 | 124 |

Figure 3.23: This table partitions sequences by their BPB scores as calculated by the grammar trained to represent them. The three middle columns show the number of sequences falling into three ranges of bits-per-base scores. A low BPB score indicates that the corresponding grammar requires little computer memory to represent that sequence.

the fundamental theorem in coding theory, which says the optimal representation of a set of strings takes $-\log(P(\text{string}))$ to represent a string; the only way a model can express a set of strings in fewer bits is to express other strings in more bits.

Figure 3.23 partitions the sequences by their BPB scores, while Figures 3.26, 3.29 and 3.32 are histograms of the BPB scores calculated by each trained grammar for its modeled set and the corresponding augmented non-snRNA set. Because of the skew in average-NLL score mentioned previously, the BPB scores for non-snRNA sequences about 30–45 nucleotides in length are artificially high, which leads to an overly long non-snRNAs tail in each of the BPB plots in Figures 3.26, 3.29 and 3.32.

### 3.3.1 U1 discrimination

The U1 grammar distinguishes the augmented non-snRNA set from the U1 set perfectly. There are no false negatives or false positives. Even the six incomplete sequences were discriminated, with Z scores ranging from 12.984 to 16.989, ranking them $4^{th}$ through $8^{th}$, and $12^{th}$ in the full set of 49 U1 sequences. The grammar was able to predict alignments for these six sequences that closely match the trusted, as well.

In contrast, the U1 initial grammar misclassifies 36 of the 49 sequences, despite its apparent ability to predict alignments that match the trusted alignments as well as those predicted by the trained grammar. As can be observed in Figure 3.25, the snRNA sequences' Z scores clearly overlap those of the non-snRNA sequences, and the snRNA Z scores are much lower in general than those given by the trained U1 grammar. Compare the histograms of Figures 3.25 and 3.24 to see the effect of grammar training on the grammar's ability to discriminate.

### 3.3.2 U5 discrimination

The U5 grammar distinguishes the augmented non-snRNA set from the U5 set nearly perfectly. The sole false negative is the incomplete sequence caimos*, 78 bases long, with the unusually low Z score of 3.858. The grammar could not align this sequence, either. However, the other incomplete sequence homsap5d* (102 bases long) scored 15.300, ranking it near the median ($13^{th}$ out of the 31 sequences), and its alignment matched the trusted in all helices but Stem II. The next lowest-scoring U5 was the training sequence schpom, 118

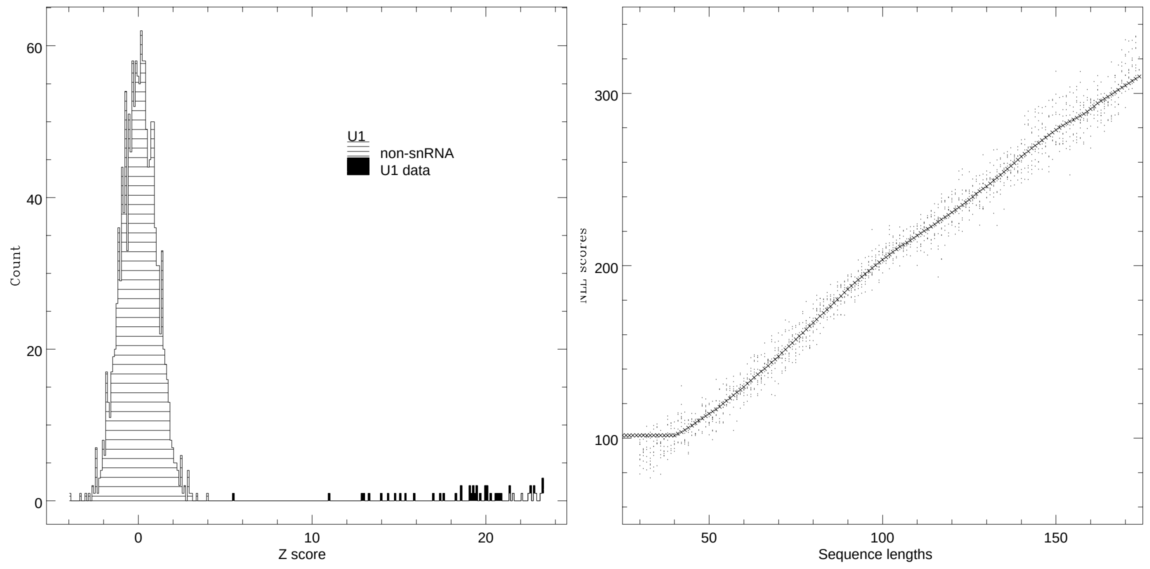


Figure 3.24: The U1 grammar was trained on 25 sequences drawn at random with equal probability from the full U1 set (Section 3.1). Though it was not trained on the test set, the trained grammar can discriminate perfectly between the entire set of U1 sequences, training plus test, and non-snRNAs (of lengths between 30 and 174) and 10 non-U1 snRNA sequences (see histogram, left). The highest-scoring augmented non-snRNA sequence is a U5 sequence 116 bases long with a Z score of 3.997, while the lowest-scoring U1 snRNA sequence has a Z score of 5.466 and is 148 bases long, leaving a margin between them about 1.5 standard deviations wide inside which a Z-scores cutoff can be chosen. NLL scores for each augmented non-snRNA sequence are also plotted (right), with the average NLL curve superimposed.

bases long, with the Z score 7.454. Thus, omitting caimos*, a Z-scores cutoff may be chosen at around 5 standard deviations. Then it is true that the U5 grammar discriminates perfectly between *complete* U5 sequences and similar-length non-snRNA fragments and other snRNA sequences.

The U5 initial grammar misclassifies 11 of the 31 U5 sequences as non-snRNAs—that is, 11 U5 sequences have Z scores below 5. As can be observed in Figure 3.28, the snRNA sequences' Z scores clearly overlap those of the non-snRNA sequences, and the snRNA Z scores are much lower in general than those given by the trained U5 grammar. Compare the histograms of Figures 3.28 and 3.27 to see the effect of grammar training on the grammar's ability to discriminate.

### 3.3.3 U6 discrimination

The trained U6 grammar discriminates complete U6 sequences from the augmented non-snRNA sequences perfectly (see Figure 3.30). However, it misclassifies both incomplete sequences, giving false negatives for leicol* and trycru*. The trained grammar's predicted alignments for these sequences deviate from the trusted alignment as well, though this is true for all 44 sequences.

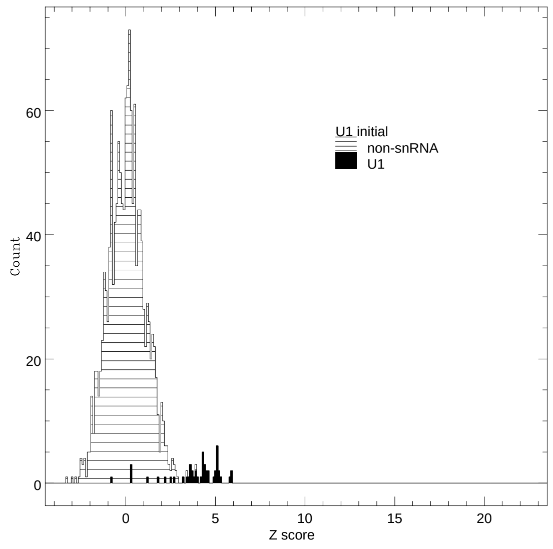


Figure 3.25: The initial U1 grammar was not trained on sequences and embodies only the Dirichlet pseudocounts. Though none of the non-snRNAs of lengths between 30 and 174 and the 10 non-U1 snRNA sequences has a Z score higher than 5, fully 36 of the 49 U1 sequences, training plus test, had Z scores lower than 5 and the highest-scoring snRNA sequence has a Z score of only 5.935. The highest-scoring augmented non-snRNA sequence is 145 bases long with a Z score of 3.865, while the lowest-scoring U1 snRNA sequence has a Z score of $-0.837$ and is 143 bases long.

As for the other trained grammars, there are no false positives. All augmented non-snRNA sequences have Z scores well below 5 standard deviations.

The U6 initial grammar misclassifies seven U6 sequences as non-snRNAs—that is, seven U5 sequences have Z scores below 5. As can be observed in Figure 3.31, the snRNA sequences' Z scores clearly overlap those of the non-snRNA sequences, and the snRNA Z scores are much lower in general than those given by the trained U6 grammar. Compare the histograms of Figures 3.31 and 3.30 to see the effect of grammar training on the grammar's ability to discriminate.

### 3.3.4 Discriminating new sequences

The grammars were used to discriminate the 32 new sequences drawn from the latest GenBank updates (82.0+, 26 May 1994, and 82.0, 8 April 1994), using the same Z-scores criterion as in previous sections: sequences scoring higher than 5 are classified as snRNA sequences. Figure 3.33 shows Z scores and BPB scores for these new sequences.

The U1 grammar succesfully discriminates 10 of the 12 new U1 snRNA sequences. The misclassified sequence `ratnor`, with a Z score of $-0.392$, seems oddly to have been aligned too far to the left relative to the other new sequences' predicted alignments, such that it appears to have an unusually long lead loop on its $5'$ end and it appears to end prematurely with no Stem IV or Loop IV. This may be an artifact from its designation in the GenBank database. The other misclassified snRNA sequence is `tetthe1`, with a Z score of 4.769.

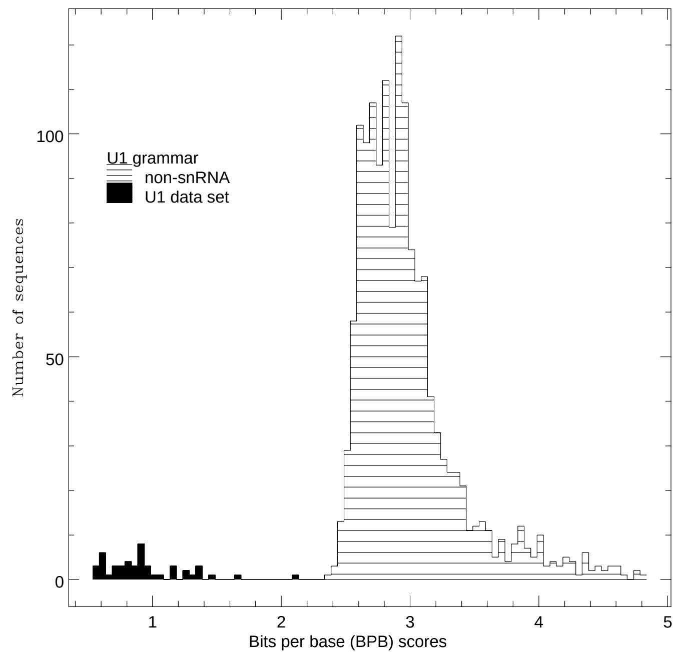


Figure 3.26: The U1 grammar was trained on 25 sequences drawn at random with equal probability from the full U1 set (Section 3.1). Shown here are the bits-per-base (BPB) scores for all 49 U1 sequences, as well as those for the 1460 augmented non-snRNA sequences. All BPB scores higher than 4 correspond to sequences shorter than 40 nucleotides in length.

The U5 grammar successfully discriminates four of the 6 new U5 snRNA sequences. The undiscriminated sequences, caeele2 and musmus, both appear to have slightly unusual alignments, with gaps inserted in some helices. These two may be incomplete sequences (fragments), they may just have unusual structure, or they may have been mislabeled in the GenBank database.

The U6 grammar successfully discriminates 10 of the 14 new U6 snRNA sequences. It misclassifies the three pseudo-U6 sequences mycoplas1, mycolplas2 and mycoplas3, ranking them lowest ($1^{st}$, $3^{rd}$ and $4^{th}$) in the new set. The trained U6 grammar does align these *Mycoplasma* sequences, which resemble U6 sequences, such that the alignments look similar to those of the actual U6 sequences, but it does not classify them as U6 sequences. This has interesting ramifications, as the *Mycoplasma* sequences come from another biological domain than the snRNAs. (See Section 4.)

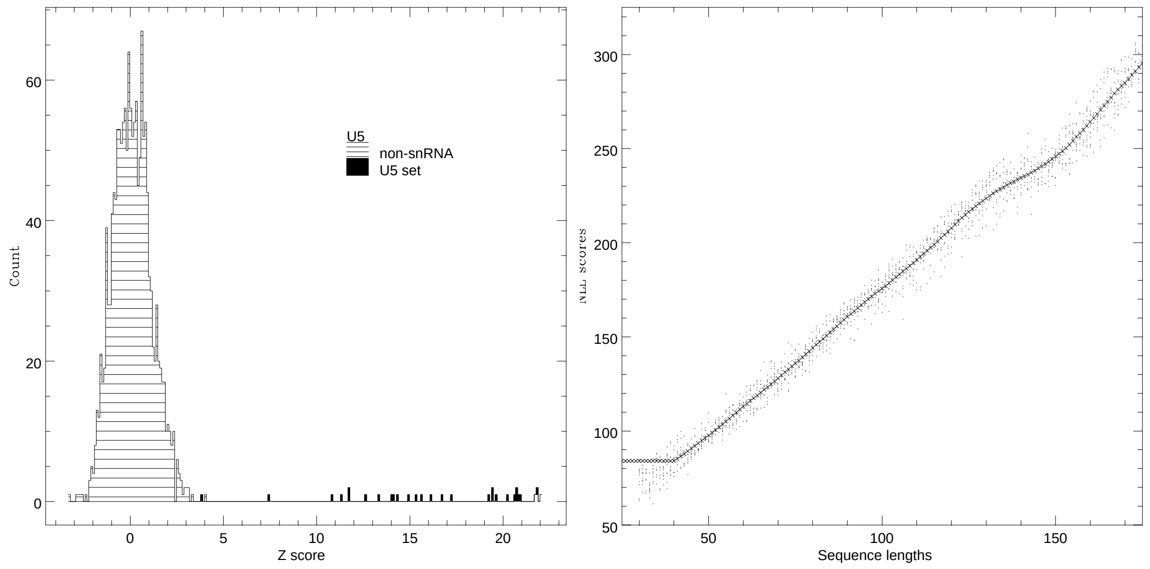


Figure 3.27: The `U5` grammar was trained on 16 complete sequences drawn at random with equal probability from the full `U5` set (Section 3.1). This histogram (left) shows discrimination between the 1460 augmented non-snRNA sequences and the entire set of `U5` sequences, training plus test. A single `U5` test sequence, the incomplete sequence `caimos*` (only 78 bases long), was misclassified as a non-snRNA sequence with a Z score of 3.858. The highest-scoring non-snRNA sequence has a Z score of 4.034 and is 106 bases long, while the next-lowest `U5` is `schpom` (118 bases long) with a Z score of 7.454, leaving a margin between them about three standard deviations wide inside which a Z-scores cutoff can be chosen. NLL scores for each augmented non-snRNA sequence are also plotted (right), with the average NLL curve superimposed.

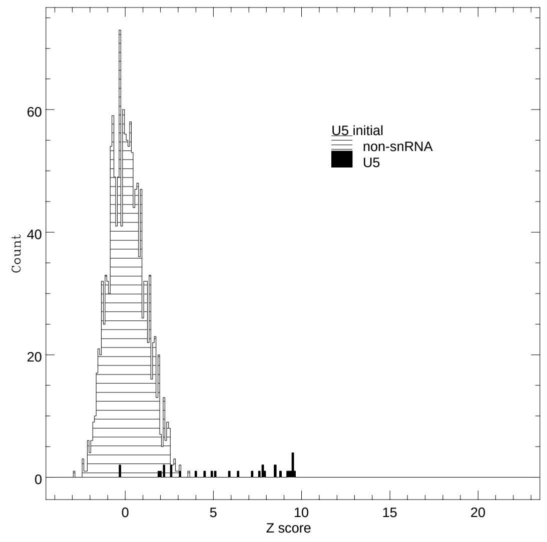


Figure 3.28: The initial `U5` grammar was not trained on sequences and embodies only the Dirichlet pseudocounts. This histogram shows its discrimination between the 1460 augmented non-snRNA sequences and the entire set of `U5` sequences, training plus test. Of the 31 total `U5` sequences, 11 were misclassified; none of the augmented non-snRNA sequences was misclassified. The misclassified `U5` snRNA sequences, with Z scores ranging from $-0.333$ to 4.869, include six `pissat` sequences, the incomplete sequence `caimos*` and the five complete sequences `aratha`, `caeele`, `schpom`, `crycoh` and `dromel`. The highest-scoring non-snRNA sequence has a Z score of 3.639 and is 105 bases long.

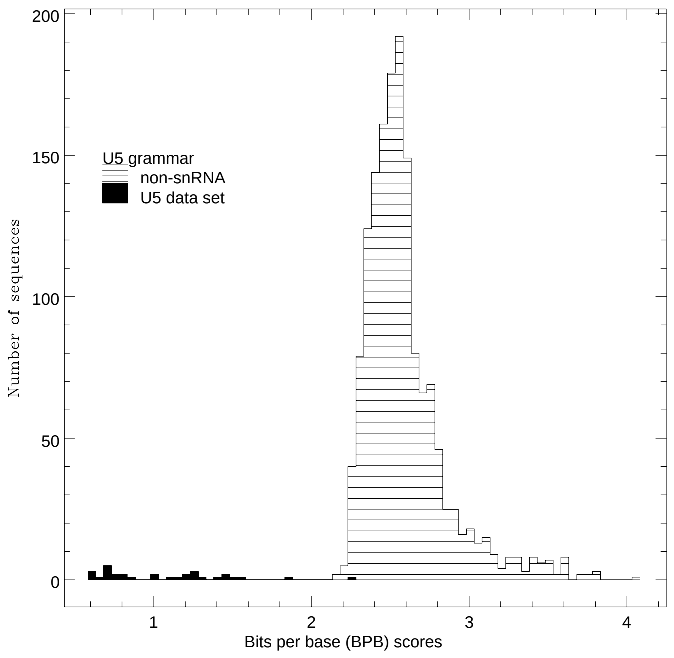


Figure 3.29: The `U5` grammar was trained on 16 sequences drawn at random with equal probability from the full `U5` set (Section 3.1). Shown here are the bits-per-base (BPB) scores for all 31 `U5` sequences, as well as those for the 1460 augmented non-snRNA sequences. All BPB scores higher than 4 correspond to sequences shorter than 40 nucleotides in length. The outlier sequence with BPB score 2.247 is the fragment `caimos*`.

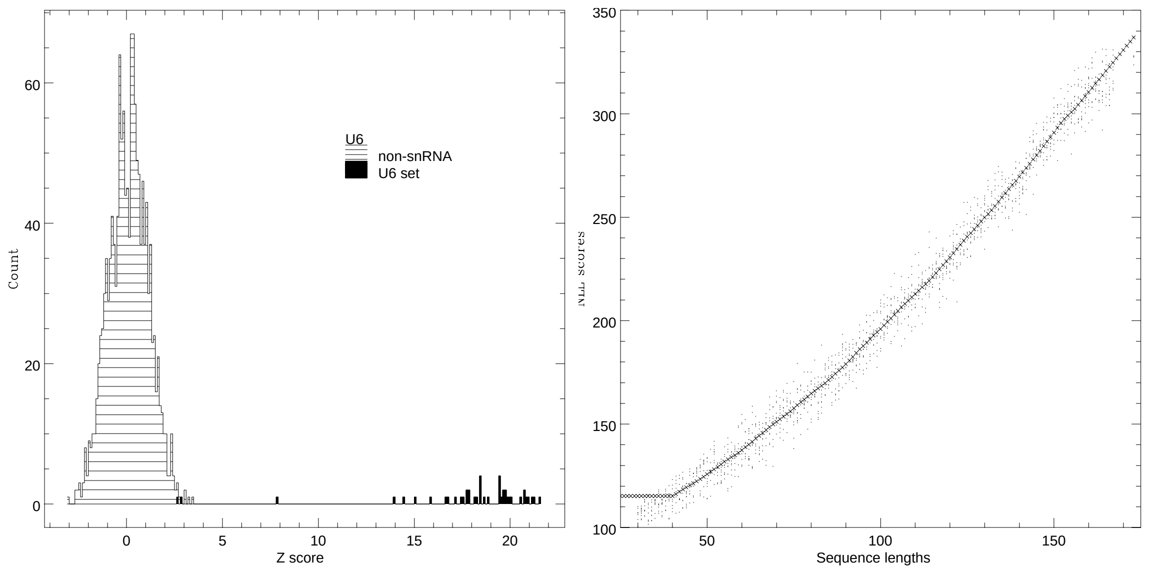


Figure 3.30: The `U6` grammar was trained on 22 complete sequences drawn at random with equal probability from the full `U6` set (Section 3.1). It discriminates nearly perfectly between the 1460 augmented non-snRNA sequences and the entire set of `U6` sequences, training plus test, though it was not trained on the test set (see histogram, left). Two `U6` test sequences, both incomplete, were misclassified as non-snRNA sequences: the lowest-scoring `U6` snRNA sequence, `trycru*`, 62 bases long, has a Z score of 2.843, while the next lowest-scoring `U6` sequence, `leicol*`, 53 bases long, has a Z score of 2.620. Ignoring these, a Z-scores cutoff can be chosen in the margin between the third lowest-scoring `U6` sequence, which is complete, 99 bases long, and has a Z score of 7.887, and the highest-scoring augmented non-snRNA sequence, 122 long, which has a Z score of 3.482. This margin is over 4 standard deviations wide. NLL scores for each augmented non-snRNA sequence are also plotted (right), with the average NLL curve superimposed.

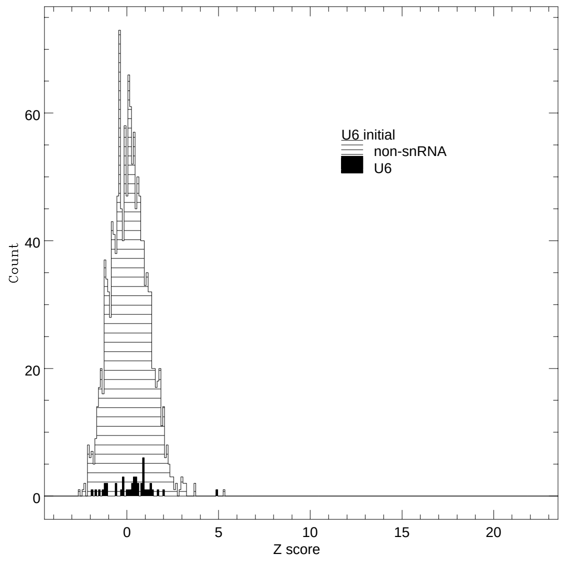


Figure 3.31: The initial U6 grammar was not trained on sequences and embodies only the Dirichlet pseudocounts. It incorrectly embeds all 44 sequences in the U6 set within the peak for the 1460 augmented non-snRNA sequences. The Z scores for the snRNA sequences range from $-1.900$ to $4.858$, while those for the augmented non-snRNA sequences range from $-2.628$ to $5.279$. There is one false positive, a non-snRNA sequence of length 111.

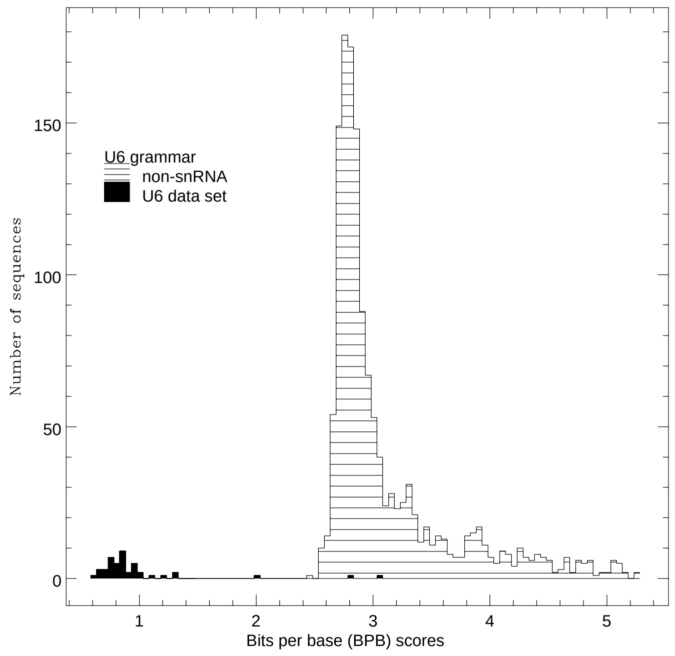


Figure 3.32: The `U6` grammar was trained on 22 sequences drawn at random with equal probability from the full `U6` set (Section 3.1). Shown here are the bits-per-base (BPB) scores for all 44 `U6` sequences, as well as those for the 1460 augmented non-snRNA sequences. All BPB scores higher than 4 correspond to sequences shorter than 40 nucleotides in length.

| Grammar | $\leq 5$ | $> 5$ | Low Zs | BPB | Name | Null BPB |
|---|---|---|---|---|---|---|
| `U1` | 2 | 10 | $-0.392$ | 2.646 | `ratnor` | 2.045 |
| | | | 4.769 | 2.194 | `tetthe1` | 2.042 |
| | | | 12.454 | 1.510 | `tetthe2` | 2.043 |
| `U5` | 2 | 4 | $-0.260$ | 2.540 | `musmus` | 2.064 |
| | | | 1.599 | 2.443 | `caeele2` | 2.072 |
| | | | 11.381 | 1.512 | `caeele1` | 2.057 |
| `U6` | 4 | 10 | $-1.011$ | 2.979 | `mycoplas2` | 2.071 |
| | | | 0.051 | 2.834 | `phypol` | 2.067 |
| | | | 1.355 | 2.660 | `mycoplas3` | 2.055 |
| | | | 1.955 | 2.605 | `mycoplas1` | 2.064 |
| | | | 7.474 | 2.035 | `trybru2` | 2.067 |
| | | | 7.505 | 2.025 | `trybru1` | 2.067 |
| | | | 14.287 | 1.295 | `tetthe` | 2.064 |
| Totals | 8 | 24 | | | | |

Figure 3.33: The 32 new snRNA sequences drawn from the latest GenBank updates were discriminated from the augmented non-snRNA set by each of the three corresponding trained grammars as shown. Most snRNA sequences had Z scores below 5. Those with Z scores above 5 also had alignments deviating substantially from those of the other new sequences in the set. BPB values are calculated as in Figure 3.22. Z scores and BPB score are shown for those sequences closest to the Z-scores cutoff.

# 4. Discussion

The SCFG method applied here represents a significant new direction in computational biosequence analysis. The results shown in this thesis, combined with recent results applying this method to modeling tRNA [SBU$^+$93, SBM$^+$94, SBH$^+$93, ED94], lend support to the analytical utility of using SCFGs to model RNA.

The aims of this work were to devise models that could provide multiple alignments, to discriminate sequences from the modeled set from sequences from other sets and to predict foldings for new sequences. This section discusses how well these goals were accomplished, compares SCFGs to other methods invented for achieving these goals and suggests possible future work to expand on the results of this thesis.

## 4.1 Assessment of results

The multiple alignments matched the trusted alignments somewhat. The results in Figures 3.7, 3.11 and 3.13 show that the trained grammars' predicted alignments do not match the trusted alignments much more closely than the initial untrained grammars, which seems to imply that training does not improve the alignment.

At least two factors may account for this. First, the snRNA sequences vary widely in their primary sequence and structure conservation, unlike the tRNA sequences upon which this method was tested previously [SBU$^+$93, SBM$^+$94, SBH$^+$93]; for instance, the helices do not have well-defined lengths. This variation, in combination with the absence of flexible bulge modeling in the grammars, may contribute to the limited precision of the alignments. Second, the figure's base-pairing match percentages were measured with respect to trusted alignments produced in reference to sequences from a single organism, the yeast *Saccharomyces cerevisiae*; snRNA sequences from other organisms were examined for motifs such as base pairs or helices that were also present in `saccer`, but cases where helices were shorter or had the possibility for a bulge were not recorded in the trusted alignments. It is possible that the trained grammars create structures custom-designed for each sequence from its modeled set on an individual basis, whereas the trusted alignments generate a generic structure, so neither predicted nor trusted alignment is truly incorrect; rather, their results may be complementary [Mia94]. It would be interesting to see whether modeling bulges such that they may appear between *any* helix-match nonterminals would greatly improve the grammars' abilities to predict alignments that closely match the trusted alignments. Many predicted snRNA sequence alignments in this work could have matched their trusted alignments exactly if a bulge position could have been moved.

Each grammar can clearly discern sequences it was intended to model from those it was not. It was shown that training on as few as about 20 sequences improves a grammar's ability to discriminate. This indicates that the grammars are limited in utility if their probabilities are derived solely from the Dirichlet prior density based on 16S rRNA.

The grammars' predicted foldings for new sequences resulted in structures closely resembling the trusted foldings in the same set. As mentioned previously, more useful grammars might result if bulges were modeled more flexibly.

The `U6` grammar folds and aligns the non-`U6` *Mycoplasma* sequences as if they were `U6` sequences, though it does not discriminate them as `U6` sequences. The genus *Mycoplasma* is bacterial while snRNAs are eucaryotic. Because, according to biologists, there appear to be three domains in life—bacteria, archaea and eucarya—the discovery of a structure or sequence that appears in two or more of these domains has important implications for understanding the evolutionary origins of domains in biological molecules. Thus a grammar that can find such structures or sequences in an RNA database would be useful to biologists. It would be interesting to experiment with grammars that model bulges more flexibly to see how they classify the *Mycoplasma* sequences and to see whether they restrict or enlarge the set of recognized sequences (those classified as a particular type of snRNA) in discrimination.

SCFGs and their equivalents *covariance models* [ED94] appear to be the sole methods currently for obtaining a single unified model that can provide RNA multiple alignments, secondary-structure predictions and sequence discrimination. Some approaches, such as thermodynamics methods, do exist for predicting helices in individual sequences, but there remains the problem of combining the results to obtain an overall picture. As yet, the focus remains on analyzing single sequences rather than families. The phylogenetic method does examine all sequences and predict a common structure, but it ignores variations in individual sequences. As mentioned in Chapter 1, HMMs are expected not to model RNA as well as SCFGs, due to the presence of base pairs in the data. It would be interesting to attempt to model snRNAs with HMMs to see what kind of discrimination and alignment results can be obtained. It is anticipated that such HMMs would lack the ability to generalize to align or discriminate new sequences.

## 4.2 Possible future tasks

SCFGs provide a flexible and highly effective statistical method for solving RNA sequence analysis problems including discrimination, multiple alignment and prediction of secondary structures. In addition, the grammar itself may be a valuable tool for representing an RNA family or domain. The present work demonstrates the usefulness of SCFGs with snRNA sequences and could prove useful in maintaining, updating and revising compilations of their alignments. Further classes of RNA sequences potentially appropriate to model using this method include group I introns [MW90, MECS90], group II introns [MUO89], ribonuclease P RNA [BHJ$^+$91, TE93] and 7S RNA (signal recognition particle RNA) [Zwi89]. Combining these models might prove useful for higher-level sequence processing. For example, a DNA database could be searched using an intron model, then the discerned intron regions could be excised and another grammar (for an snRNA set, for example) could then be used to parse the remaining regions to find new RNA sequences.

The main difficulties in applying this work to other families of RNA are developing appropriate initial grammars and decreasing the computation time required to parse longer sequences. Even for these relatively short snRNA sequences (53–170 bases in length) and grammars with 1498–2021 productions and 367–498 nonterminals, the required memory makes parsing nearly intractable on current architectures. The latter problem can be solved only by the development of fundamentally different parsing methods, perhaps relying more on branch-and-bound methods (a form of best-first search) [LS94] or heuristics. It is currently not clear which approach will be best. The former problem might be solved by the development of effective methods for learning the grammar itself from training sequences. The work of Eddy and Durbin is an important step in this direction [ED94]. Their method relies on correlations between columns in a multiple alignment [GPH$^+$92, Lap92, KB93, Wat89, WOW$^+$90, San85, Wat88] to discover the essential base-pairing structure in an RNA family. Another approach would be to use a method like that proposed by Waterman

[Wat89] to find helices in a rough initial multiple alignment, use these helices to design a simple initial grammar in a semi-automated fashion using the high-level RNA grammar specification language discussed in Section 2.5, then use the grammar to obtain a better multiple alignment, and iterate this process until a suitable result is obtained. Recently some researchers have used Gibbs sampling strategies [GG84] for producing multiple alignments and deducing locations of helices from raw sequences [GHH$^+$94, LAW$^+$94, LAB$^+$93, BL93]. The algorithm of Lawrence *et al.* [LAW$^+$94, LAB$^+$93] for proteins produces local multiple alignments by finding blocks of residues conserved across sequences. The algorithm of Grate *et al.* [GHH$^+$94] extends the Lawrence approach to produce local multiple alignments of RNA. Because RNA secondary structure must be taken into account to produce these, the Grate approach finds conserved based pairs (i.e., helices). Another approach that might reduce computation time would be to use much simpler grammars that do not rely on having fixed maximum helix lengths. By including recursive productions of the form $S \rightarrow aSb$, a helix of any length could be modeled in principle. Such a grammar might model secondary structure better but model primary structure worse. This tack would decrease the number of productions and thus the amount of memory required, but might complicate the alignment algorithm.

Because shorter sequences require less time to parse, and because biologists often work with fragments rather than complete sequences, a related goal would be to devise schemes for modeling sequence fragments. These snRNA grammars failed to align or discriminate sequences that were only 50–70% the length of the average sequence length in the corresponding set, while length deviations of up to 40% from the average length seemed to be accommodated. It might be useful to characterize how short a fragment can become before a grammar begins to misclassify it, but the results would depend on the other sequences present in the set.

Another important direction for further research is the development of stochastic grammars for snRNA and other RNA families that can be used to search databases for these structures at the DNA level. In order to do this, the grammar must be modified to allow for the possibility of introns in the sequence, and the parsing method must be modified so that it can efficiently search for RNAs that are embedded within larger sequences. Durbin and Eddy have done the latter modifications in their tRNA experiments and report good results in searching the GenBank structural RNA database and 2.2 Mb of *C. elegans* genomic sequence for tRNAs, even without using special intron models. Earlier work [SBM$^+$94] reported some very preliminary results on modifying tRNA grammars to accommodate introns. Though it should be straightforward to develop effective stochastic grammar-based search methods, the main practical problem will be dealing with the long computation time required by the present methods.

Finally, there is the question of what further generalizations of hidden Markov models, beyond SCFGs, might be useful. The key advantage of the SCFG method over the HMM method is that it allows explicit treatment of the secondary structure of the RNA sequence. By extending stochastic models of strings to stochastic models of trees, the base-pairing interactions of the molecule, which determine its secondary structure, can be modeled. This progression is similar to the path taken by the late King Sun Fu and colleagues in their development of the field of syntactic pattern recognition [Fu82]. Modeling pseudoknots and higher-order structure would require still more general methods. One possibility would be to consider *stochastic graph grammars* (see the introductory survey by Engelfriet and Rozenberg [ER91]) in hopes of obtaining a more general model of the interactions present in the molecule beyond the primary structure. If a stochastic graph grammar framework could be developed that included both an efficient method of finding the most probable folding of the molecule given the grammar and an efficient EM method for estimating the grammar's parameters from folded examples, then extensions of this approach to more challenging problems, including RNA tertiary structure determination and protein folding, would be possible. This is perhaps the most interesting direction for future research encouraged by the results of this thesis.

# Appendix A. Predicted Multiple Alignments

Each section in this appendix shows the complete multiple alignment of an snRNA set (training plus test sequences) as predicted by the corresponding trained grammar. They agree to varying degrees with the trusted alignments for these same sequences (see Sections 3.2.1, 3.2.2 and 3.2.3).

Periods demark helix boundaries and also indicate application of skip productions. In the `U5` and `U6` multiple alignments, the lowercase letters denote application of an insert production. In the `U1` alignment, the loops are not aligned. Bulges were modeled as a loop between two short helices for Stems I, III and IV in `U1` and for Stem I (broken into Stems Ia, Ib and Ic) in `U5`.

The snRNA sequences come from a variety of organisms, including yeast (`saccer`), pea (`pissat`), human (`homsap`), fruit fly (`dromel`), rat (`ratnor`), mouse (`musmus`), slime mold (`phypol`), and chicken (`galgal`). Sequences whose names are appended with an asterisk `*` are incomplete.

## A.1 Full `U1` predicted alignment

```
                         Trunk   Stem I w/Bulge              Stem I        Stem II
                         (((((( (((((     (((((            )))))))))) (((((((((((((((
schpom      ----ACUUACCU.GGCAUG.AGUUU.CUG.CAGCA.---CAAGAAU.UGUGGAGACU.CAGUUAUUUGUCUUGG.--CAUUGCAC-UG
chlsac      --AUACUUACCU.GUCCGG.CCUGC.-G-.ACCUC.GAGCAAGAAG.GGGGUCUAGG.UAGUGCUUGUACCUC-.GCCUUG-UAC-UA
glymax1A    --AUACUUACCU.GGACGG.GGUCA.-A-.UGGAU.GAUCAAGAAG.GUCCAUGGCC.UAGGGAAGUAACCUCC.AUUGCACUGAG--
glymax1B    --AUACUUACCU.GGACGG.GGUCA.-A-.UGGAU.GAUCAAUAAG.GUCCAUGGCC.UAGGGAAGUAACCUCC.AUUGCACUUAG--
lycesc      --AUACUUACCU.GGACGG.GGUCA.-A-.UGGGC.GAUCAAUAAG.ACCCAUGGCC.UAGGCUUGUGACCUCC.AUUGCACUUUG--
lycesc12    --AUACUUACCU.GGACGG.GGUCA.-U-.UGGGC.GAUCAAUAAG.ACCCAUGGCC.UAGGCUUGUGACCUCC.AUUGCACUUCG--
lycesc13    --AUACUUACCU.GGACGG.GGUCA.-A-.UAGGC.GAUCAAUAAG.ACCCAUGGCC.UAGGUUGGUGACCUCC.AUUGCACUUUG--
lycesc14    --UUACUUACCU.GGACGG.GGUCA.-A-.UUGGC.GAUCAUGAAG.GUCCAUGGCC.UAGGUUGGUAACCUCC.AUUGCACUUAG--
lycesc15    --AUACUUACCU.GGACGG.GGUCU.-A-.UGGGC.GAUCAUGUAG.GUCCAUGGCC.UAGGUUGGUGACUUUC.AUUGCACUUUG--
lycesc16    --AUACUUACCU.GGACGG.GGUCA.-A-.UGGGU.AAUCAAGAAG.UUCCAUGGCC.UAGGUUGGUGACCUCC.AUUGCACUAAG--
lycesc17    --AUACUUACCU.GGACGG.GGUCA.-A-.UGGGC.GAUCAUUAAG.ACCCAUGGCC.UAGGCUUGUGACCUCC.AUUGCACUUUG--
lycesc18    --AUACUUACCU.GGACGG.GGUCA.-A-.UGGGC.GAUCAUGAAC.ACCCAUGGCC.UAGGUUGGUGACCAUC.AUUGCACUUUG--
pissat1A    --AUACUUACCU.GGAUGG.GGUC-.-GA.UGGGU.GAUCAUGAAG.GCCCA-UGGC.UAGGAUUGUGACCUCC.AUUGCACUUAG--
pissat1B    --AUACUUACCU.GGAUGG.GGUC-.-AA.UGGGU.GAUCAAGAAG.GCCCA-UGGC.UAGGCAAGUGACCUCC.AUUGCACUUAG--
pissat1C    --AUACUUACCU.GGAUGG.GGU-C.-AA.UGGGU.GAUCAUGAAG.GCCCA-UGGC.UAGGCAAGUGACUUCC.AUUGCACUUAG--
phavul      --AUACUUACCU.GGACGG.GGUCA.-A-.UGGAU.GAUCUAUAAG.GUCCAUGGCC.UAGGGAAGUGACCUUC.AUUGCACUCAG--
triaes11    --AUACUUACCU.GGACGG.GGU-C.-GA.CGGGC.GAUCAAGAAG.GCCCG-UGGC.UGGGUCAAUGGCUCAC.AUUGCACUUGG--
triaes12    ----ACUUACCU.GGACGG.GGUCG.-A-.CGGCC.GAUCAAGAAG.GGUCGUGGCC.UAGAUCAAUGGUCACA.-UUGCACCUGG--
triaes13*   ------UUACCU.GGACGG.GGUCG.-A-.CGAGC.GAUCAAGAAG.GCUCGUGGCC.UAGGUUAGUGGCCCAC.AUUGCACUUGG--
triaes14*   --AUACUUACCU.GGACGG.GGUCG.-A-.CGAGC.GAUCAAGAAG.GCUCGUGGCC.UAGGUUAGUGGCCCAC.AUUGCACUUGG--
triaes15*   --AUACUUACCU.GGACGG.GGUCG.-A-.CGAGC.GAUCAAGAAG.GCUCGUGGCC.UGGGUCAGUGGUCCAC.AUUGCACUU-G--
triaes16*   ----ACUUACCU.GGACGG.GGU-C.-GA.CGGGC.GAUCAAGAAG.GCCCG-UGGC.UGGGUCGAUGGCCCAC.AUUGCACUUGG--
tetthe      ----ACUUACCU.GGCUGG.AGUUU.-G-.CUAUC.GAUCAUGAAG.GGUAGCGGCU.UAGGGUGGAGCAGGUC.AUUGCACAAAAGA
lytvar1G    --AUACUUACCU.GGCGCA.GGGGU.-C-.GCAUU.GAUCAAGAAG.GAUGCACCCC.CAGGGCGAGGCUU-GC.CAUUGCACUCCG-
lytvar2G    --AUACUUACCU.GGCGCA.GGGGU.-C-.GCAUU.GAUCAAGAAG.GAUGCACCCC.CAGGGCGAGGCUU-GC.CAUUGCACUCCG-
stropur1G   --AUACUUACCU.GGCGCA.GGGGU.-C-.GCAUU.GAUCAAGAAG.GAUGCACCCC.CAGGGCGAGGCUU-GC.CAUUGCACUCCG-
echmul      --AAACUUACCU.GGCGCC.GGGUU.-C-.AGGGU.GAUCAGCAAG.GCUCCGACCC.CAGGUGGAGGCUCAG-.CAUUGCACUCCG-
caeele1     AUAAACUUACCU.GGCUGG.GGGUU.-AU.UUCGC.GAUCACAAAG.GCGGAAUCCC.CAUGGUUAGGCCUACC.CAUUGCACUUUUG
dromel19523 --AUACUUACCU.GGCGUA.GAGGU.-UA.ACCGU.GAUCACGAAG.GCGGUUCCUC.CGGAGUGAGGCUUGGC.CAUUGCACCUCG-
dromelG1A   --AUACUUACCU.GGCGUA.GAGGU.-UA.ACCGU.GAUCACGAAG.GCGGUUCCUC.CGGAGUGAGGCUUGGC.CAUUGCACCUCG-
ambmex      --AUACUUACCU.GGCAGG.GGAGC.-A-.UCUGU.GAUCAGCAAG.GCAGAGCUCC.CAGGGUGAGGCUCAUC.CAUUGCACAUCG-
xenlae15    --AUACUUACCU.GGCAGG.GGAGA.-U-.ACCAU.GAUCAUGAAG.GUGGUUCUCC.CAGGGCGAGGCUCAGC.CAUUGCACUCCG-
xenlae1ABG  --AUACUUACCU.GGCAGG.CGAGA.-U-.ACCAU.GAUCACGAAG.GUGGUUCUCC.CAGGGCGAGGCUCAGC.CAUUGCACUCCG-
xenlaeABG   --AUACUUACCU.GGCAGG.GGAGA.-U-.ACCAU.GAUCACGAAG.GUGGUUCUCC.CAGGGCGAGGCUCAGC.CAUUGCACUCCG-
xenlae1D1*  --------ACCA.UGAUCA.CGA--.-A-.GGU--.G--------G.--UUC--UCC.CAGGGCGAGGCUCAGC.CAUUGCACUCCG-
xenlae1D2*  -----------U.----CC.-----.---.-----.----------.----------.CAGGGCGAGGCUCAGC.CAUUGCACUCCG-
galgal1R    --AUACUUACCU.GGCAGG.GGAGA.-C-.ACCAU.GAUCAGGCAG.GUGGUUUUCC.CAGGGCGAGGCUCAUC.CCCUGCACUCCG-
ratnor4     ----ACUUACCU.GGCAGG.GGAGA.-U-.ACCAU.GAUCACGAAG.GUGGUUUUCC.CAGGGCGAGGCUUAUC.CAUUGCACUCCG-
ratnor1183A --AUACUUACCU.GGCAGG.GGAGA.-U-.ACCAU.GAUCACGAAG.GUGGUUUUCC.CAGGGCGAGGCUUAUC.CAUUGCACUCCG-
```

```
                         Trunk  Stem I w/Bulge              Stem I        Stem II
                         (((((( (((((     (((((            ))))))))) ((((((((((((((((
ratnor1183B --AUACUUACCU.GGCAGG.GGAGA.-U-.ACCGU.GAUCACGAAG.GUGGUUUCC.CAGGGCGAGGCUUAUC.CAUUGUACUCCG-
musmus1B2   --AUACUUACCU.GGCA-G.GGGAG.-AU.ACCAU.GAUCAUGAAG.GUGGUUUCC.CAGGGCGAGGCUCACC.CAUUUGCACUGUU
musmus1B2A  --AUACUUACCU.GGCAGG.GGAGA.-U-.ACCAU.GAUCAUGAAG.GUGGUUUCC.CAGGGCGAGGCUCACC.CAUUGCACUUUGG
musmusG1A   --AUACUUACCU.GGCAGG.GGAGA.-U-.ACCAU.GAUCACGAAG.GUGGUUUCC.CAGGGCGAGGUGUAUC.CAUUGCA--UCCG
musmusR1A   --AUACUUACCU.GGCAGG.GGAGA.-U-.ACCAU.GAUCACGAAG.GUGGUUUCC.CAGGGCGAGGCUUAUC.CAUUGCACUCCG-
musmus1B6   --AUACUUACCU.GGCAGG.GGAGA.-U-.ACCAU.GAUCACGAAG.GUGGUUUCC.CAGGGCGAGGCUCACC.CAUUGCACUUUGG
bostau      --AUACUUACCU.GGCAGG.GGAGA.-U-.ACCAU.GAUCACGAAG.GUGGUUUCC.CAGGGCGAGGCUUAUC.CAUUGCACUCCG-
homsap1A    --AUACUUACCU.GGCAGG.GGAGA.-U-.ACCAU.GAUCACGAAG.GUGGUUUCC.CAGGGCGA-GGCUUAU.CCAUUGCACUCCG
homsap1C    --AUACUUACCU.GGCAGG.GGAGA.-U-.ACCAU.GAUCACGAAG.GUGGUUUCC.CAGGGCGAGGCUUAUC.CAUUGCACUCCG-

                   Stem II      Stem III          StemIII w/Blg  Trunk                       StemIV B
               )))))))))))))))) (((((((((         ))))))    ))) ))))))                        (((((
schpom      AG.CCCUGACGAAUAACUG.UGGACUGGC.-UAA-G-.GUCAGC.U-.CCG.GAUGCA.U---------C---A-U-----.-UUUU.G
chlsac      UG.-CUUGGGGUAGCGCUG.UGUGCGGGG.-CAA-GU.CCUCGU.U-.ACA.ACGGAA.UAAUUUCUGGC---AGG-----.CCGUU.G
glymax1A    --.GAGGGGUGCCUUUCUA.AGGUCUGUC.-CAA-GU.GACAGA.G-.CCU.ACGUCA.UAAUUUGUGGUA----G-----.UGGGG.G
glymax1B    --.GAGGGGUGCUUUCCUA.AGGUCUGCC.-CAA-GU.GGCAGA.G-.CCU.ACGUCA.UAAUUUGUGGUAGU--------.GGGGG.C
lycesc      --.GAGGGGUGCCUGCCUA.AGGUCGGCU.-CAA-GU.AGUCGA.G-.CCU.ACGUCA.UAAUUUGUUGCAGA--------.GGGGG.C
lycesc12    --.GAGGGGUGCUUGUCUA.AGGUCGGCU.-CAA-GC.AGUCGA.G-.CCU.ACGUCA.UAAUUUGUUGCA-G--------.UGGGG.G
lycesc13    --.GAGGGGUGCCAAUCUA.AGGUCGGCC.-CAA-GU.GGUCGA.G-.CCU.ACGUCA.UAAUUUGUUGCUGU--------.GAGGG.C
lycesc14    --.GAGAGGUGCCUACCUA.AGAUCGGCC.-CAA-GU.GGCCGA.A-.UCU.ACGUCA.UAAUUUGUUGCUGA--------.GGGGG.C
lycesc15    --.GAGGGGUGCCCGCCUA.AGAUCAGCC.-CAA-GA.GGUUGA.G-.UCU.ACAUCA.UAAUUUGUUGCUGA--------.GGGGG.C
lycesc16    --.GAGGGGUGCUUGCCUA.AGGUCGACC.-CAA-GU.GGUUGA.G-.CCU.ACGUCA.UAAUUUGUUGUUGCAGA-----.GGGGG.C
lycesc17    --.GAGGGGUGCCUGCCUA.AGGUUGGCC.-CAA-GU.GGUCGA.G-.CCU.ACGUCA.UAAUUUGUUGCUGU--------.GGGGG.C
lycesc18    --.AAGGGGUGCCCGCCUA.AGGUCGGCC.-CAA-GU.GGUCGA.G-.CCU.ACGUCA.UAAUUUGUUGCUGA--------.GGGGG.C
pissat1A    --.GAGGGGUGCUUUCCUA.AGGUCUACC.-CAA-GU.GGUGGA.G-.CCU.ACAUCA.UAAUUUGUUGCCUG--------.AGGGG.G
pissat1B    --.GAGGGGUGCUAGCCUA.AGGUCUACC.-CAA-GU.GGUGGA.G-.CCU.ACAUCA.UAAUUUGUUGCUGU--------.GGGG-.G
pissat1C    --.GAGGGGUGCUUGCCUA.AGGUCUACC.-CAA-GU.GGUGGA.A-.CCU.ACAUCA.UAAUUUGUUGCUGU--------.GGGGG.C
phavul      --.AAGGGGUGCUACUCUA.AGGUCUGUC.-CAA-GU.GAUGGA.G-.CCU.ACGUCA.UAAUUUGUGGUAGU--------.GGGGG.C
triaes11    --.UGGGUGCGUUGGCCCA.CCAUCUCCC.-CAA-GU.GGGAGA.G-.UGG.ACGUCA.UAAUUUGUGCUA-G--------.AGGGG.G
triaes12    --.UGAGCGCGUUGGCCUA.CCAUCUCCC.-CAA-GU.GGGAGA.G-.UGG.ACGUCG.UAAUUUGUGGUA-G--------.AGGGG.G
triaes13*   --.UGGGUGCGCUGGCCUA.UCAUCUCCC.-CAA-GU.GGGAGA.G-.UGA.ACGUCA.UAAUUUGUGGUA-G--------.AGGGG.G
triaes14*   --.UGGGUGCGCUGGCCUA.UC
triaes15*   --.GUGGAUGCCUGGCCCA.CCAUCUCCC.-CAA-GU.GGGAGA.G-.UGG.AUGUCA.UAAUUUGUGGUAGAGGGGGUAC
triaes16*   --.UGGGUGCGUCGGCCCA.UCA
tetthe      --.UGUCUGUAAUACCUUA.UUGUUCCCC.---GUGC.GGGGAA.CC.GAA.ACAGCA.CAAUUUCUGCUAGG--------.GGAGA.C
lytvar1G    --.GC-UUGCUGAACCUUG.CGAUUCCCC.CAAACGU.GGGGAA.C-.UCG.GGCGUA.CAAUUUAUGAUAGC--------.GGAGA.U
lytvar2G    --.GC-UUGCUGAACCUUG.CGAUUCCCC.CAAACGU.GGGGAA.C-.UCG.GGCGUA.CAAUUUAUGGUAGC--------.GGAGA.U
stropur1G   --.GC-UUGCUGAACCUUG.CGAUUCCCC.CAAACGU.GGGGAA.C-.UCG.GGCGUA.UUAUUUAUGGUAGC--------.GGAGA.U
echmul      --.-CUGUGUUGAAGCCUG.CGACGGACU.CUAAUCG.GGUUCG.C-.UCG.GGUGCA.UAGUUUUUGC---C--------.AGUGG.G
caeele1     GG.UGCGGGCUGACCUGUG.UGGCAGUCU.CGAGUUG.AGAUUC.G-.CCA.ACAGCU.UAAUUUUUGCGUA-U-------.CGGG-.G
dromel19523 --.GCUGAGUUGACCUCUG.CGAUUAUUC.CUAAUGU.GAAUAA.C-.UCG.UGCGCG.UAAUUUUUGGUAGC--------.CGGG-.A
dromelG1A   --.GCUGAGUUGACCUCUG.CGAUUAUUC.CUAAUGU.GAAUAA.C-.UCG.UGCGUG.UAAUUUUUGGUAGC--------.CGGG-.A
ambmex      --.GAUUUGCUGACCCCUG.CGAUGUCCC.CAAAUGC.GGGAUU.C-.UCG.ACUGUA.UUAUUUCUGGUAGU--------.GGGG-.G
xenlae15    --.GUUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGC.GGGAAA.C-.UCG.ACUGCA.UAAUUUCUGGUAGU--------.GGGG-.G
xenlae1ABG  --.GCUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGC.GGGAAA.C-.UCG.ACUGCA.UAAUUUCUGGUAGU--------.GGGG-.G
xenlaeABG   --.GCCGUGCUGACCCCUG.CGAUUUCCC.CAAAUGC.GGGAAA.G-.UCG.ACUGCA.UAAUUUCUGGUA-G--------.UGGGG.G
xenlae1D1*  --.GCCGUGCUGACCCCUG.CGAUUUCCC.CAAAUGC.GGGAAA.G-.UCG.ACUGCA.UAAUUUCUGGUA-G--------.UGGGG.G
xenlae1D2*  --.GCUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGU.GGGAAA.C-.UCG.AC----.UGCAUAAUUUGUGGUA-G----.UGGGG.G
galgal1R    --.GGUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGC.GGGAAA.C-.UCG.ACUGCA.UAAUUUGUGGUA-G--------.UGGGG.G
ratnor4     --.GAUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGC.GGGAAA.C-.UCG.ACUGCA.UAAUUUGUGGUAGU--------.GGGG-.G
ratnor1183A --.GAUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGC.GGGAAA.C-.UCG.ACUGCA.UAAUUUGUGGUA-G--------.UGGGG.G
ratnor1183B --.GAUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGC.GGGAAA.C-.UCG.ACUGCA.UAAUUUGUGGUA-G--------.UGGGG.G
musmus1B2   -G.GGUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGC.GGGAAA.C-.UCG.A-UGCA.-AAUUUGUGGUA-G--------.UGGGG.G
musmus1B2A  --.GGUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGC.GGGAAA.C-.UCG.ACUGCA.UAAUUUGUGGUA-G--------.UGGGG.G
musmusG1A   --.GAUGUGCUGACCACUG.CGAUUUCCC.CAAAUGC.GGGAAA.C-.UCG.ACUGCA.UAAUUUGUGGUA-G--------.UGGGG.G
musmusR1A   --.GAUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGC.GGGAAA.C-.UCG.ACUGCA.UAAUUUGUGGUAGU--------.GGGG-.G
musmus1B6   --.GGUGUGCUGACCCCUG.CGAUUUCCC.-AAAUGC.GGGAAA.C-.UCG.ACUGCA.UAAUUUGUGGUAGU--------.GGGGG.A
bostau      --.GAUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGU.GGGAAA.C-.UCG.ACUGCA.UAAUUUGUGGUAGU--------.GGGGG.A
homsap1A    --.GAUGUGC-UGCCCCUG.CGAUUUCCC.CAAAUGU.GGGAAA.C-.UCG.ACUGCA.UAAUUUGUGGUA-G--------.UGGGG.G
homsap1C    --.GAUGUGCUGACCCCUG.CGAUUUCCC.CAAAUGU.GGGAAA.C-.UCG.ACUGCA.UAAUUUGUGGUA-G--------.UGGGG.G
```

```
            StemIV        Stem IV
            ((((((      )))))))))))
schpom      AGUUCG.UCCC.UCAUUUGGGG-.-CA-
chlsac      CACGCG.CUUG.CGCGUCCUCGG.cAA-
glymax1A    CCUGCG.UUCG.CGCGGCCCCUU.-UC-
glymax1B    -UUGCG.UUCG.CGCAG-CCCCU.-UC-
lycesc      -CUGCG.UUCG.CGCAG-CCCCU.--A-
lycesc12    CCUGCG.UUCG.CGCAGCCCCUA.-UC-
lycesc13    -CUGUG.UUCG.CGCGG-CCCCU.-GC-
lycesc14    -CUGCG.UUCG.CGCGG-CCCCU.-GC-
lycesc15    -AUGCG.UUCG.CGCAG-CCCCU.-GC-
lycesc16    -CUGUG.UUCG.CGCAG-CCCCU.-AC-
lycesc17    -CUGCG.UACG.CGCAG-CCCCU.-GC-
lycesc18    -CUGCG.UUCG.CGCGG-CCCCU.--G-
pissat1A    CCUGCG.UUCG.CGCGGCCCCCA.-CC-
pissat1B    CCUGUG.UUCG.CGCGG-CCCCC.-UC-
pissat1C    -CUGCG.UUCG.CGCGG-CCCUC.-UU-
phavul      -CUGCG.UUCG.CGCGG-CCCCU.-UA-
triaes11    UACGCG.UUCG.CGCGGCCCCUG.-CU-
triaes12    UACGCG.UUCG.CGCGGCCCCUA.-CU-
triaes13*   UACGCG.UUCG.CGCGGCCCCUG.-CU-
tetthe      GUGCAC.UU-A.GUGCUGUCUCC.GCUC
lytvar1G    C-UGCG.UUCG.CGCU-AUCUCC.-UA-
lytvar2G    C-UGCG.UUCG.CGCU-AUCUCC.-UA-
stropur1G   C-UGCG.UUCG.CGCU-AUCUCC.-UA-
echmul      GA-GCC.UUCG.GGC-GUCCCUU.-UC-
caeele1     -CUGCG.UGCG.CGCGG--CCCU.-GA-
dromel19523 AUGGCG.UUCG.CGCCGU-CCCG.--A-
dromelG1A   AUGGCG.UUCG.CGCCGU-CCCG.--A-
ambmex      ACUGCG.UUCG.CGCUUU-CCCC.-UG-
xenlae15    ACUGCG.UUCG.CGCUUU-CCCC.-UG-
xenlae1ABG  ACUGCG.UUCG.CGCUUU-CCCC.-UG-
xenlaeABG   ACUGCG.UUCG.CGCUUUCCCCU.-GA-
xenlae1D1*  ACUGCG.UUCG.CGCUUUCCCCU.-GU-
xenlae1D2*  ACUGCG.UUCG.CGCUUUCCCCU.-GA-
galgal1R    ACUGCG.UUCG.CGCUCUCCCCU.-GA-
ratnor4     ACUGCG.UUCG.CGCUCU-CCCC.-UG-
ratnor1183A ACUGCG.UUCG.CGCUCUCCCCU.-GA-
ratnor1183B ACUGCG.UUCG.CGCUCUCCCCU.-GG-
musmus1B2   ACUGCG.UUCG.CGCUCUCCCCU.-GA-
musmus1B2A  ACUGCG.UUCG.CGCUCUCCCCU.-GA-
musmusG1A   ACUGCG.UUUG.UGCUCUCCCCU.-UU-
musmusR1A   ACUGCG.UUCG.CGCUCU-CCCC.-UG-
musmus1B6   GCUGCG.UUCG.CGCGCCCCCCU.-GUA
bostau      ACUGCG.UUCG.CGCUUU-CCCC.-UG-
homsap1A    ACUGCG.UUCG.CGCUUUCCCCU.-GA-
homsap1C    ACUGCG.UUCG.CGCUUUCCCCU.-GA-
```

## A.2 Full U5 multiple alignment

```
                                                         Ia       Stem Ib                    Stem Ic
                                                        ((((    (((((((((((                 ((((((((
schpom     ...Auaauccg................................U.CAAA.-G.CACUUUGCAAA.AgcuA.A...CGUAU.CUGUUUCU
aratha     ...Gggaguaaaaaucacgc.......................A.GCCA.UG.UGGUGAGUACA.A...A.G...CGAAC.UAUUUCUU
pissat5a1  gg.A.......................................G.CCGU.-G.UGAUGAUGACA.U...A.G...CGAAC.UAUUCUUU
pissat5b1  gg.A.......................................G.CCAU.-G.UGAUAAGUACA.A...A.G...CGAAC.UAUUCUUU
pissat5a2  gg.A.......................................G.CCGU.-G.UGAUGAUGACA.U...A.G...CGAAC.UAUCUUUC
pissat5b2  ...G.......................................G.AGCCgUG.UGAUGAACACA.A...A.G...CGAAC.UAUCUUUC
pissat5c   gg.A.......................................G.CCGU.-G.UGAUGAUGACA.U...A.G...CGAAC.UAUCUUUC
pissat5e   gggA.......................................G.CCAU.-G.UGAUAAGUGCA.A...A.G...CGAAC.UAUCUUUC
pissat5f   gggA.......................................G.CCAU.-G.UGAUAAGUGCA.A...A.G...CGAAC.UAUCUUUC
crycoh     ...G.......................................A.UCAC.AG.UGUUCACUUCA.-...A.C...CGAAU.CAAUCUUU
tetpyr5a   ...A.......................................U.CACA.-G.AACUCAGCUCA.A...U.A...CGCUU.UAAUUUUU
tetthe51   ...A.......................................U.CACA.-G.AACUCAGCUCA.U...U.A...CGCUU.UAAUUUUU
caeele     ...Aauc....................................A.ACUC.UG.GUUCCUCUGCA.U...U.UaacCGUGA.AAAUCUUU
dromel     ...A.......................................U.ACUC.UG.GUUUCUCUUCA.A...UgU...CGAAU.AAAUCUUU
```

```
                                                         Ia       Stem Ib                  Stem Ic
                                                        ((((    (((((((((((                ((((((((
xenlae     ...A.......................................U.ACUC.UG.GUUUCUCUUCA.A...AuU...CGAAU.AAAUCUUU
galgal5arn ...A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGUAU.AAAUCUUU
galgal5a   ...A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGUAU.AAAUCUUU
galgal5b   ...A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGUAU.AAAUCUUU
caimos*    ...CcuuuuacuaaagauuuccguggagaggaacaaccacgagU.----GUC.GUGG.A.AU...UUUUUGAGGCUCCGC.......U
musmus5    g..A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGUAU.AAAUCUUU
ratnor5a1  ...A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGUAU.AAAUCUUU
ratnor5a2  g..A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGUAU.AAAUCUUU
ratnor5ax  ...A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGUAU.AAAUCUUU
homsap5b   ...A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGUAU.AAAUCUUU
homsap5a1  ...A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGCAU.AAAUCUUU
homsap5a2  ...A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGCAU.AAAUCUUU
homsap5b1  ...A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGCAU.AAAUCUUU
homsap5b1s ...A.......................................U.ACUC.UG.GUUUCUCUUCA.G...A.U...CGUAU.AAAUCUUU
homsap5d*  ...A.......................................U.GCUC.UG.GUUUCUCUUCA.A...A.U...CGUAU.AAAUCUUU
homsap5e   ...A.......................................U.ACUC.UG.GUUUCUCUUCA.A...A.U...CGUAU.AAAUCUUU
homsap5f   ...A.......................................A.UCUC.UG.GUUUCUCUUCA.U...A.A...CGAAU.AAAUCUUU

                        Stem Ic        Stem Ib              Ia                         Stem II
                        ))))))))     )))))))))))           ))))                        (((((((
schpom     .UGCCUUUUACC.AGAAACAG.CCG.UUUGUAAGGUG.U.G.CU.AA.UUUG.ACUGUA-.U.AG..UUUUUGAA.UCUU---......
aratha     uCGCCUUUUACU.AAAGAAUA.CCG.UGUGCUCUCGA.C.G.CU.AA.GUGC.AUACGCC.U.AU..UUUUGGAG.GGCUCCA.cuuc.
pissat5a1  .CGCCUUUUACU.AAAGAAUA.CCG.UGUCAGCGUCA.C.A.AU.UA.GCGG.CAUACGC.U.AG..UUUUUGGA.AGAGUUC.ucaau
pissat5b1  .CGCCUUUUACU.AAAGAAUA.CCG.UGUACUUUUCA.C.U.AA.CA.GUGG.CAUACGA.U.AA..UUUUUGAA.UGAGCUC.ucaug
pissat5a2  .-GCCUUUUACU.AAAGAAUA.CUG.UGUCAGCGUCA.C.A.AU.UA.GCGG.CAUACGC.U.AG..UUUUUGGA.AGAGUUC.ucaag
pissat5b2  .-GCCUUUUACU.AAAGAAUA.CUG.UGUGUGCGUCA.C.U.AA.AA.GGCG.CAUACGC.CuAA..UUUUUGAA.AGAGUUC.ucuu.
pissat5c   .-GCCUUUUACU.AAAGAAUA.CUG.UGUCAGCGUCA.C.A.AU.UA.GCGG.CAUACGC.U.AG..UUUUUGGG.AGAGUCC.ucuuc
pissat5e   .-GCCUUUUACU.AAAGAAUA.CUG.UGUACGUGUCA.C.A.AG.CG.GUGG.CAUACGAgU.AA..UUUUUGAA.UGAGUUC......
pissat5f   .-GCCUUUUACU.AAAGAAUA.CUG.UGUACGUGUCA.C.A.AG.CG.GUGG.CAUACGAgU.AA..UUUUUGAA.UGAGUUC......
crycoh     .CGCCUUUUACU.AAAGGUUG.CCG.UGAAUGGGACA.C.AuCA.AU.GUGA.AUCUCUC.A.AU..UU-----U.UGAG-GG......
tetpyr5a   .CGCCUUUUACU.AAAGAUUA.CCG.UGGGCUGGGUU.C.U.AC.AA.UGUG.AAUUAUU.A.AA..AUUUUUGA.GGAUUGU......
tetthe51   .CGCCUUUUACU.AAAGAUUA.CCG.UGGGCUGGGUU.U.A.CC.AA.UGUG.AAUUAUU.A.AA..AUUUUUGC.AGGAUUC......
caeele     .CGCCUUUUACU.AAAGAUUU.CCG.UGCAAAGGAGC.A.U.UUaCU.GAGU.AUUACAU.A.CA..AUUUUUGG.AGACUCC......
dromel     .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGGAAC.AcU.CU.AA.GAGU.CUAAAAC.U.AA..UUUUUUAG.UCAGUCU......
xenlae     .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGGAAC.G.A.CC.AU.GAGU.UUCGUUC.A.AU..UUUUUGAA.GCCUGGU......
galgal5arn .CGCCUUUCAUC.AAAGAUUU.CCG.UGGAGAGGAAC.A.A.CU.CU.GAGU.CUUAAAC.C.AA..UUUUUUGA.GCCUUGU......
galgal5a   .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGGAAC.A.A.CC.AC.GAGU.GUCGUGG.A.AU..UUUUUGAG.GCUCCGC......
galgal5b   .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGGAAC.A.A.CC.AC.GAGU.GUCGUGG.A.AU..UUUUUGAG.GCUCCGC......
caimos*    UC......GACG.GAGC.....U--.--
musmus5    .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGGAAC.A.A.CU.CU.GAGU.CUUAAAC.C.AA..UUUUUUGA.GGCCUUG.uc...
ratnor5a1  .CGCCUUUCAUC.AAAGAUUU.CCG.UGGAGAGGAAC.A.A.CU.CU.GAGU.CUUAAAC.C.AA..UUUUUUGA.GCCUUGU......
ratnor5a2  .CGCCUUUUAUC.AAAGAUUU.CCG.UGGAGAGGAAC.A.A.CU.CU.GAGU.CUUAAAC.C.AA..UUUUUUGA.GGCCUUG.uc...
ratnor5ax  .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGGAAC.A.A.CU.CU.GAGU.CUUAAAC.C.AA..UUUUUUGA.GCCUUGU......
homsap5b   .CGCCUUUCAUC.AAAGAUUU.CCG.UGGAGAGGAAU.A.A.CU.CU.GAGU.CUUAAGC.U.AA..UUUUUUGA.GCCUUGC......
homsap5a1  .CGCCUUUCAUC.AAAGAUUU.CCG.UGGAGAGGAAC.A.A.CU.CU.GAGU.CUUAACC.C.AA..UUUUUUGA.GCCUUGC......
homsap5a2  .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGGAAC.A.A.CU.CU.GAGU.CUUAACC.C.AA..UUUUUUGA.GCCUUGC......
homsap5b1  .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGGAAC.A.A.CU.CU.GAGU.CUUAACC.C.AA..UUUUUUGA.GCCUUGC......
homsap5b1s .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGGAAU.A.A.CU.CU.GAGU.CUUAAGC.U.AA..UUUUUUGA.GCCUUGC......
homsap5d*  .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGAAAC.C.G.UU.UU.GAGU.UUCUAGC.U.AA..UUUUUUGA.--AG---......
homsap5e   .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGAAAC.G.A.GU.GU.GAGU.CUGAAAC.C.AA..UUUUUUGA.GGCCUUG.ccuuu
homsap5f   .CGCCUUUUACU.AAAGAUUU.CCG.UGGAGAGAAAC.A.A.CU.AU.GAGU.UUAUGGU.U.AAauUUUUUGAA.GUCUUGC......

                        Stem II
                        )))))))
schpom     ..UUU......C.---UUGA.a...A
aratha     ..UCU......G.UGGAACC.ca..A
pissat5a1  u.UUG......A.GGGCUCU.....G
pissat5b1  u.UUG......A.GAGCUCU.....G
pissat5a2  uuUUG......A.GGGCUCU.....G
pissat5b2  ..UUG......A.GAGCUCU.g...G
pissat5c   a.UUG......A.GGGCUCU.....G
pissat5e   ..UCUuguuagA.GAACUCU.....G
pissat5f   ..UCUuguuagA.GAGCUCU.....A
crycoh     ..CUC......U.GC-CCCA.....C
tetpyr5a   ..---......G.UGAAUCC.u...A
```

```
                       Stem II
                       )))))))
tetthe51   ..UUU......U.GAAUCCU.cucaC
caeele     ..UUGagaaa.G.CGGGUCA.aa..A
dromel     ..UGUcgc...A.AGACUGG.ggccA
xenlae     ..UCA......C.ACCAGGU.....A
galgal5arn ..UCC......G.GCAAGGC.u...A
galgal5a   ..UUC......G.ACGGAGC.....U
galgal5b   ..UUC......G.GCGGAGC.u...A
musmus5    ..UUG......G.CAAGGCU.....A
ratnor5a1  ..UCC......G.GCAAGGC.ua..A
ratnor5a2  ..UUG......A.CAAGGCU.....A
ratnor5ax  ..UCC......G.ACAAGGC.ua..A
homsap5b   ..UCC......G.ACAAGGC.ua..A
homsap5a1  ..CUU......G.GCAAGGC.ua..A
homsap5a2  ..CUU......G.GCAAGGC.ua..A
homsap5b1  ..UCC......G.ACAAGGC.ua..A
homsap5b1s ..UCC......G.ACAAGGC.ua..A
homsap5d*  ..---......C.---CC--.....U
homsap5e   u.UUA......G.CAGGGCU.....U
homsap5f   ..CUA......G.GCAAGGC.uc..G
```

## A.3 Full U6 multiple alignment

```
                      ((((((((                      ))))))))
saccer     ..GuU....C.GCGAAGUA...AC.......C.......C.UUCGUGGA.....CAUUUGG.UCAAUUUGAAAC.AAU...AC
sacdai     ..-.-....C.CCGGAGUA...AC.......C.......C.UUCGUGGA.....CAUUUGG.UCAAUUUGAAAC.AAU...AC
sacexi     ..-.U....C.GCGAAGUA...AC.......C.......C.UUCGUGGA.....CAUUUGG.UCAAUUUGAAAC.AAU...AC
sacklu     ..-.-....C.CAAGACAU...UU.......C.......G.GUGUUUUG.ga..CAUUUGG.UCAAUUUGAAAC.AAU...AC
sacser     ..-.-....C.UUCCCGGA.uuAA.......C.......G.UCCGUGGA.a...CAUUUGG.UCAAUUUGAAAC.AAU...AC
sactel     ..-.-....U.UUCCCGGA...--uuag...C.......G.UCCGUGGA.a...CAUUUGG.UCAAUUUGAAAC.AAU...AC
sacbay     ..-.U....C.GCGAAGUA...AC.......C.......C.UUCGUGGA.....CAUUUGG.UCAAUUUGAAAC.AAU...AC
saccas     ..-.C....C.CCAAGGCA...AC.......C.......C.CUCGUGGA.....CAUUUGG.UCAAUUUGAAAC.AAU...AC
sacuni     ..U.U....C.CCGGAUUA...-A.......C.......G.UCCGUGGA.a...CAUUUGG.UCAAUUUGAAAC.AAU...AC
klulac     ..-.A....G.UCCAUAUU...AC.......CcuccgguU.UUUAUGGA.....CAUUUGG.UCAAUUUGAAAC.AAU...AC
piccan     ..-.C....C.CUUGUGGA...CA.......U.......U.UGGUCAAU.....-------.-----UAGAAAU.AAU...AC
picgui     ..C.C....C.CUUGUGGA...CA.......U.......U.UGGUCAAU.....-------.-----UAGAAAU.AAU...AC
pichee     ..-.-....C.CUCGGGGA...CA.......U.......A.UGGUCAAA.....C------.-------GAAAA.AAU...AC
picmis     ..-.-....C.CUUAUGGA...CA.......U.......U.UGGUCAAU.....-------.-----UAGAAAU.AAU...AC
pactan     ..-.-....C.CUCG-GGA...CA.......U.......U.UAG-UCAA.....-------.----UUUGAAAC.AAU...AC
saccap     ..-.-....C.-CGA-GGA...CC.......C.......C.UCG-UGG-.a...CAUAUGG.UCAAUUUGAAAC.AAU...AC
sacfib     ..-.-....C.UUCG-GGA...CA.......U.......U.UGG-UCAA.....-------.----UUUGUUACaAAU...AC
yarlip     ..-.-....C.UUCG-GGA...CA.......U.......A.UGG-UCAA.....-------.----UUUGAAAA.AAU...AC
nadful     ..-.-....C.CCAUUCGU...--.......-.......G.GGACAUAU.....-----GG.UCAAUUUGAAAC.GAU...AC
saclud     ..-.-....U.CGGAGCGU...A-.......-.......A.GCUCUGGA.....CAUUUGGuUAAAUUUGAAAC.AAU...AC
zygflo     ..-.-....C.GUGGAGUA...AC.......C.......C.UUCAUGGA.....CAUUUGG.UCAAUUUGAAAC.AAU...AC
zygrou     ..-.C....C.GUGGAGUA...AC.......C.......C.UUCAUGGA.....CAUUUGG.UCAAUUUGAAAC.AAU...AC
tordel     ..-.-....C.CCGAAGUA...AC.......C.......C.UUCGUGGA.....CAUUUGG.UCAAUUUGAAAC.AAU...AC
ambmon     ..-.-....C.UCUUCGGA...GG.......C.......A.UUUAGUUA.....-------.----AUCGAAAA.AAU...AC
lipsta     ..U.U....C.--------...--.......-.......-.-------g.ggaCAUAUGG.UCAAUUUGAAAC.GAU...ACA
issori     ..-.-....U.UCUUCGGA...CAgcgugguC.......A.AGCGAAGA.....-------.------------.AAU...AC
schpom     ..-.-....G.AUGA-UCU...--.......U.......C.GGA-UCAC.....--UUUGG.UCAAAUUGAAAC.GAU...AC
aratha61   ..G.U....C.CCUUCGGG...GA.......C.......A.UCCGAUAA.....-------.---AAUUGGAAC.GAU...AC
aratha626  ..G.U....C.CCUUCGGG...GA.......C.......A.UCCGAUAA.....-------.---AAUUGGAAC.GAU...AC
aratha629  ..G.U....C.CCUUCGGG...GA.......C.......A.UCCGAUAA.....-------.---AAUUGGAAC.GAU...AC
zeamay     ..G.U....C.UCUUCGGA...GA.......C.......A.UCCGAUAA.....-------.---AAUUGGAAC.GAU...AC
vicfab     ..-.-....C.--UUCGGG...GA.......C.......A.UCCGAU--.....-------.-AAAAUUGGAAC.GAC...AC
lycesc     ..G.U....C.CCUUCGGG...GA.......C.......A.UCCGAUAA.....-------.---AAUUGGAAC.GAU...AC
trybru     ..-.-ggagC.CCUUCGGG...GA.......C.......A.UCCACAAA.....-------.-----CUGGAA-.-AUucaAC
trycru*    ..G.A....G.CCUUCGUG...UC.......C.......A.AGCGAAGG.....-ACAUCC.ACAAUCUGGAA-.-AUucaAC
leicol*    gaG.C....C.CUUCGGGG...GA.......C.......A.UCCACAAC.....C------.-----UGGAAAC.UCA...AC
caeele     ..G.U....U.CUUCCGAG...AA.......C.......A.UAUACUAA.....-------.---AAUUGGAAC.AAU...AC
dromel1    ..-.-....G.UUCUUGCU...--.......-.......U.CGGCAGAA.....CAUAUAC.UAAAAUUGGAAC.GAU...AC
dromel2    ..-.-....G.UUCUUGCU...--.......-.......U.CGGCAGAA.....CAUAUAC.UAAAAUUGGAAC.GAU...AC
dromel3    ..-.-....G.UUCUUGCU...--.......-.......U.CGGCAGAA.....CAUAUAC.UAAAAUUGGAAC.GAU...AC
```

```
                      ((((((((                      ))))))))
xentro     ..-.-....G.UGCUUGCU..--.......-........U.CGGCAGCA....CAUAUAC.UAAAAUUGGAAC.GAU...AC
musmus     ..-.-....G.UGCUCGCU..--.......-........U.CGGCAGCA....CAUAUAC.UAAAAUUGGAAC.GAU...AC
musmus6A   ..-.-....G.UGCUUGCU..--.......-........U.CGGCAGCA....CAUAUAC.UAAAAUUGGAAC.GAU...AC
homsap     ..-.-....G.UGCUCGCU..--.......-........U.CGGCAGCA....CAUAUAC.UAAAAUUGGAAC.GAU...AC

saccer     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUAUUU...CG.UUUU
sacdai     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUAUUC...GU.UU-U
sacexi     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUAUUU...CG.UUUU
sacklu     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUACUU...CA.UUUU
sacser     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUAUAC...AU.UU-U
sactel     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUACACgu.UU.UUUU
sacbay     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUAUUUc..GU.UUUU
saccas     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUAUUC...GU.UUUU
sacuni     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUAUAC...AU.UUUU
klulac     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUA--A...GA.UUUU
piccan     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUACUUa..UU.UUUU
picgui     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUACUA...--.UUUU
pichee     AGAGAAGAUUAGCAUGGCCCCUGCAUAAGGAUGACAC.G...CU.UU.ACAAAGAGAUc.UUACUC...AAcUUUU
picmis     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UG.UU.ACAAAGAGAU..UUACAA...--.UUUU
pactan     AGAGAAGAUUAGCAUGGCCCCUGCAUAAGGAUGACAC.G...CU.UU.ACAAAGAGAU..UAAACC...AU.UUUU
saccap     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UAAUUC...AU.UUUU
sacfib     AGAGAAGAUUAGCAUGGCCCCUGCAUAAGGAUGACAC.G...CU.UU.ACAAAGAGAU..UUAACC...GU.UUUU
yarlip     AGAGAAGAUUAGCAUGGCCCCUGCAUAAGGAUGACAC.G...CU.UU.ACAAAGAGAU..UUAAACc..GU.UUUU
nadful     AGAGAAGAUUAGCAUGGCCCCUGCAUAAGGAUGACAC.G...CU.UU.ACAAAGAGAU..UUACAA...GU.UUUU
saclud     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUUAAU...AU.UUUU
zygflo     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUACACg..UU.UUUU
zygrou     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UAAUUCau.UU.UUUU
tordel     AGAGAUGAUCAGCAGUUCCCCUGCAUAAGGAUGAACC.G...UU.UU.ACAAAGAGAU..UUAUUG...AU.UUUU
ambmon     AGAGAAGAUUAGCAUGGCCCCUGCAUAAGGAUGACAC.G...CU.UU.ACAAAGAGAUcuUUAUUAcu.UU.UUUU
lipsta     GAGAAGAUUAGCAUGGCCCCUGCAUAAGGAUGACAC.G...CU.UUgAUAAAGAGAG..UAUGAU...GU.UUUU
issori     AGAGAAGAUUAGCAUGGCCCCUGCAUAAGGAUGACAC.G...CU.UU.ACAAAGAGAUc.UUACUC...AAcUUUU
schpom     AGAGAAGAUUAGCAUGGCCCCUGCACAAGGAUGACACuG...CG.AC.AUUGAGAGA-..-AAACC...CA.UUUU
aratha61   AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CAuAA.AUCGAGAAAU..GGUCCAaauUU.UUUU
aratha626  AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CAuAA.AUCGAGAAAU..GGUCCAaauUU.UUUU
aratha629  AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CAuAA.AUCGAGAAAU..GGUCCAaauUU.UUUU
zeamay     AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CAcAA.AUCGAGAAAU..GGUCCAaauUU.UUUU
vicfab     AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CAcAA.AUCGAGAAAU..GGUCCA...AA.UUUU
lycesc     AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CAcAA.AUCGAGAAAU..GGUCCAaaaUU.UUUU
trybru     AGAGAAGAUUAGCACUCUCCCUGCGCAAGGCUGA---.-uguCA.AUcUUCGAGAGAU..AUAGC-...--.UUUU
trycru*    AGAGAAGAUUAGCA-----------------------.-...--.--.----------..------...--.---C
leicol*    AGAGAAGAUUAGCA-----------------------.-...--.--.----------..------...--.---C
caeele     AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CA.AA.UUCGUGAAGC..GUUCCAa..AU.UUUU
dromel1    AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CA.AA.AUCGUGAAGC..GUUCCAc..AU.UUUU
dromel2    AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CA.AA.AUCGUGAAGC..GUUCCAc..AU.UUUU
dromel3    AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CA.AA.AUCGUGAAGC..GUUCCA...CA.UUUU
xentro     AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CA.AA.UUCGUGAAGC..GUUCCAu..AU.UUUU
musmus     AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CA.AA.UUCGUGAAGC..GUUCCA...UA.UUUU
musmus6A   AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CA.AA.UUCGUGAAGC..GUUCCA...UA.UUUU
homsap     AGAGAAGAUUAGCAUGGCCCCUGCGCAAGGAUGACAC.G...CA.AA.UUCGUGAAGC..GUUCCA...UA.UUUU
```

## References

[Arn93] F. H. Arnold. Engineering proteins for nonnatural environments. *Faseb Journal*, 7:744–749, 1993.

[AU72] A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation and Compiling, Vol. I: Parsing.* Prentice Hall, Englewood Cliffs, N.J., 1972.

[AWM+84] R. M. Abarbanel, P. R. Wieneke, E. Mansfield, D. A. Jaffe, and D. L. Brutlag. Rapid searches for complex patterns in biological molecules. *Nucleic Acids Research*, 12(1):263–280, 1984.

[Bak79] J. K. Baker. Trainable grammars for speech recognition. *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550, 1979.

[BBB+84] R. Baer, A. T. Bankier, M. D. Biggin, P. L. Deininger, P. J. Farrell, T. J. Gibson, G. Hatfull, G. S. Hudson, S. C. Satchwell, and C. Seguin. DNA sequence and expression of the B95-8 Epstein-Barr virus genome. *Nature*, 310:207–211, 1984.

[BBH93] J. M. Burke and A. Berzal-Herranz. *In vitro* selection and evolution of RNA: applications for catalytic RNA, molecular recognition, and drug discovery. *Faseb Journal*, 7:106–112, 1993.

[BCHM93] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure. Hidden Markov models in molecular biology: new algorithms and applications. In Hanson, Cowan, and Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 747–754, San Mateo, CA, 1993. Morgan Kauffmann Publishers.

[Ber92] B. Berkhout. Structural features in TAR RNA of human and simian immunodeficiency viruses: a phylogenetic analysis. *Nucleic Acids Research*, 20:27–31, 1992.

[BHJ+91] J. W. Brown, E. S. Haas, B. D. James, D. A. Hunt, J. S. Liu, and N. R. Pace. Phylogenetic analysis and evolution of RNase P RNA in proteobacteria. *Journal of Bacteriology*, 173:3855–3863, 1991.

[BHK+93] M. P. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjölander, and D. Haussler. Using Dirichlet mixture priors to derive hidden Markov models for protein families. In L. Hunter, D. Searls, and J. Shavlik, editors, *Proc. of First Int. Conf. on Intelligent Systems for Molecular Biology*, pages 47–55, Menlo Park, CA, July 1993. AAAI/MIT Press.

[BKM+81] C. Branlant, A. Krol, M. A. Machatt, J. Pouyet, J. P. Ebel, K. Edwards, and H. Kossel. Primary and secondary structures of *Escherichia coli* MRE 600 23S ribosomal RNA. Comparison with models of secondary structure for maize chloroplast 23S rRNA and for large portions of mouse and human 16S mitochondrial rRNAs. *Nucleic Acids Research*, 9:4303–4324, 1981.

[BL93] S. H. Bryant and C. E. Lawrence. An empirical energy function for threading protein sequence through the folding motif. *Proteins: Structure, function, and genetics*, 16:92–112, 1993.

[CAKF86] F. E. Cohen, R. M. Abarbanel, I. D. Kuntz, and R. J. Fletterick. Turn prediction in proteins using a pattern-matching approach. *Biochemistry*, 25:266–276, 1986.

[CC93] R. H. Crozier and Crozier Y. C. The mitochondrial genome of the honeybee *Apis mellifera*: complete sequence and genome organization. *Genetics*, 133:97–117, 1993.

[Chu89] G. A. Churchill. Stochastic models for heterogeneous DNA sequences. *Bull Math Biol*, 51:79–94, 1989.

[CK91] D. K. Chiu and T. Kolodziejczak. Inferring consensus structure from nucleic acid sequences. *Computer Applications in the Biosciences*, 7:347–352, 1991.

[CMDM90] D. J. Cummings, K. L. McNally, J. M. Domenico, and E. T. Matsuura. The complete DNA sequence of the mitochondrial genome of *Podospora anserina*. *Current Genetics*, 17:375–402, 1990.

[Cou91] J. Courteau. Genome databases. *Science*, 254:201–207, 1991.

[CRR$^+$89] P. Cantatore, M. Roberti, G. Rainaldi, M. N. Gadaleta, and C. Saccone. The complete nucleotide sequence, gene organization, and genetic code of the mitochondrial genome of *Paracentrotus lividus*. *Journal of Biological Chemistry*, 264:10965–10975, 1989.

[CS92] L. R. Cardon and G. D. Stormo. Expectation maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragments. *Journal of Molecular Biology*, 223:159–170, 1992.

[Cur79] Helena Curtis. *Biology*. Worth Publishers, New York, New York, third edition, 1979.

[CZJ91] L. Chan, M. Zuker, and A. B. Jacobson. A computer method for finding common base paired helices in aligned sequences: application to the analysis of random sequences. *Nucleic Acids Research*, 19:353–358, 1991.

[DA89] J. E. Dahlberg and J. N. Abelson, editors. *RNA processing. Part A. General Methods* , volume 180 of *Methods in Enzymology*. Academic Press, New York, 1989.

[Doo90] R. F. Doolittle, editor. *Molecular Evolution*, volume 183 of *Methods in Enzymology*. Academic Press, New York, 1990.

[DPBB92] D. L. Daniels, G. Plunkett 3d, V. Burland, and F. R. Blattner. Analysis of the *Escherichia coli* genome. DNA sequence of the region from 84.5 to 86.5 minutes. *Science*, 257:771–778, 1992. E. coli sequencing project.

[DS81] J. J. Dunn and F. W. Studier. Nucleotide sequence from the genetic left end of bacteriophage T7 DNA to the beginning of gene 4. *Journal of Molecular Biology*, 148:303–330, 1981.

[DS83] J. J. Dunn and F. W. Studier. Complete nucleotide sequence of bacteriophage T7 DNA and the locations of T7 genetic elements. *Journal of Molecular Biology*, 166:477–535, 1983.

[DSC83] D. L. Daniels, F. Sanger, and A. R. Coulson. Features of bacteriophage lambda: analysis of the complete nucleotide sequence. *Cold Spring Harbor Symposia on Quantitative Biology*, 47:1009–1024, 1983.

[ED94] S. R. Eddy and R. Durbin. RNA sequence analysis using covariance models. *NAR* in press, 1994.

[ER91] J. Engelfriet and G. Rozenberg. Graph grammars based on node rewriting: An introduction to NLC graph grammars. In E. Ehrig, H.J. Kreowski, and G. Rozenberg, editors, *Lecture Notes in Computer Science*, volume 532, pages 12–23. Springer-Verlag, 1991.

[FB91] G. A. Fichant and C. Burks. Identifying potential tRNA genes in genomic DNA sequences. *Journal of Molecular Biology*, 220:659–671, 1991.

[Fu82] K. S. Fu. *Syntactic pattern recognition and applications.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[FW75] G. E. Fox and C. R Woese. 5S RNA secondary structure. *Nature*, 256:505–507, 1975.

[GG84] S. Geman and D. Geman. Stochastic relaxations, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6:721–742, 1984.

[GHH$^+$94] L. Grate, M. Herbster, R. Hughey, I.S. Mian, H. Noller, and D. Haussler. RNA modeling using Gibbs sampling and stochastic context free grammars. In *Proc. of Second Int. Conf. on Intelligent Systems for Molecular Biology*, Menlo Park, CA, August 1994. AAAI/MIT Press.

[GMC90] D. Gautheret, F. Major, and R. Cedergren. Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA. *Computer Applications in the Biosciences*, 6(4):325–331, 1990.

[Gou] M. Gouy. Secondary structure prediction of RNA. pages 259–284.

[GP88] C. Guthrie and B. Patterson. Spliceosomal snRNAs. *Annual Review of Genetics*, 22:387–419, 1988.

[GPD$^+$89] G. Gadaleta, G. Pepe, G. De Candia, C. Quagliariello, E. Sbisa, and C. Saccone. The complete nucleotide sequence of the *Rattus norvegicus* mitochondrial genome: cryptic signals revealed by comparative analysis between vertebrates. *Journal of Molecular Evolution*, 28:497–516, 1989.

[GPH$^+$92] R. R. Gutell, A. Power, G. Z. Hertz, E. J. Putz, and G. D. Stormo. Identifying constraints on the higher-order structure of RNA: continued development and application of comparative sequence analysis methods. *NAR*, 20:5785–5795, 1992.

[GRM93] C. Guthrie, H. Roha, and I. S. Mian. Spliceosomal snRNA structure: function. Unpublished, 1993.

[GZB81] C. Glotz, C. Zwieb, and R. Brimacombe. Secondary structure of the large subunit ribosomal RNA from *Escherichia coli*, *Zea mays* chloroplast, and human and mouse mitochondrial ribosomes. *Nucleic Acids Research*, 9:3287–3306, 1981.

[HAE$^+$65] R. W. Holley, J. Apgar, G. A. Everett, J. T. Madison, , M. Marquisee, S. H. Merrill, J. R. Penswick, and A. Zamir. Structure of a ribonucleic acid. *Science*, 147:1462–1465, 1965.

[HHD$^+$93] R. B. Hallick, L. Hong, R. G. Drager, M. R. Favreau, A. Monfort, B. Orsat, A. Spielmann, and E. Stutz. Complete sequence of *Euglena gracilis* chloroplast DNA. *Nucleic Acids Research*, 21:3537–3544, 1993.

[HK93] K. Han and H.-J. Kim. Prediction of common folding structures of homologous RNAs. *Nucleic Acids Research*, 21:1251–1257, 1993.

[HKMS93] D. Haussler, A. Krogh, I. S. Mian, and K. Sjölander. Protein modeling using hidden Markov models: Analysis of globins. In *Proceedings of the Hawaii International Conference on System Sciences*, volume 1, pages 792–802, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[HSS93] L. Hunter, D. Searls, and J. Shavlik, editors. *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology.* AAAI/MIT Press, Menlo Park, CA, July 1993.

[HSW$^+$89] J. Hiratsuka, H. Shimada, R. Whittier, T. Ishibashi, M. Sakamoto, M. Mori, C. Kondo, Y. Honji, C.-R. Sun, B.-Y. Meng, Y.-Q. Li, A. Kanno, Y. Nishizawa, A. Hirai, K. Shinozaki, and M. Sugiura. The complete sequence of the rice (*Oryza sativa*) chloroplast genome: intermolecular recombination between distinct tRNA genes accounts for a major plastid DNA inversion during the evolution of the cereals. *Molecular and General Genetics*, 217:185–194, 1989.

[Hun92] Lawrence Hunter. *Molecular Biology for Computer Scientists*, chapter 1, pages 1–46. AAAI Press, 1992.

[Joi93] Joint NIH/DOE Mouse Working Group. A plan for the mouse genome project. *Mammalian Genome*, 4(6):293–300, 1993.

[JOP89] B. D. James, G. J. Olsen, and N. R. Pace. Phylogenetic comparative analysis of RNA secondary structure. *Methods in Enzymology*, 180:227–239, 1989.

[JTZ90] J. A. Jaeger, D. H. Turner, and M. Zuker. Predicting optimal and suboptimal secondary structure for RNA. *Methods in Enzymology*, 183:281–306, 1990.

[KB93] T. Klinger and D. Brutlag. Detection of correlations in tRNA sequences with structural implications. In Lawrence Hunter, David Searls, and Jude Shavlik, editors, *First International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, 1993. AAAI Press.

[KBM$^+$94] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, Feb. 1994.

[KMH93] A. Krogh, I. S. Mian, and D. Haussler. A Hidden Markov Model that finds genes in *E. coli* DNA. Technical Report UCSC-CRL-93-33, University of California at Santa Cruz, Computer and Information Sciences Dept., Santa Cruz, CA 95064, 1993. in preparation.

[LAB$^+$93] C. E. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald, and J. Wootton. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, 1993.

[Lap92] Allan Lapedes. Private communication, 1992.

[LAW$^+$94] C. E. Lawrence, S. Altschul, J. Wootton, M. Boguski, A. Neuwald, and J. Liu. A Gibbs sampler for the detection of subtle motifs in multiple sequences. In *Proceedings of the Hawaii International Conference on System Sciences*, pages 245–254, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[Lev69] M. Levitt. Detailed molecular model for transfer ribonucleic acid. *Nature*, 224:759–763, 1969.

[LG87] E. S. Lander and P. Green. Construction of multilocus genetic linkage maps in humans. *Proceedings of the National Academy of Sciences of the United States of America*, 84:2363–2367, 1987.

[LOM$^+$93] N. Larsen, G. J. Olsen, B. L. Maidak, M. J. McCaughey, R. Overbeek, T. J. Macke, T. L. Marsh, and C. R. Woese. The ribosomal database project. *Nucleic Acids Research*, 21:3021–3023, 1993.

[LS94] R. H. Lathrop and T. F. Smith. A branch-and-bound algorithm for optimal protein threading with pairwise (contact potential) amino acid interactions. In *Proceedings of the 27th Hawaii International Conference on System Sciences*, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[LWS87] R. H. Lathrop, T. A. Webster, and T. F. Smith. Ariadne: pattern-directed inference and hierarchical abstraction in protein structure recognition. *Communications of the ACM*, 30(11):909–921, 1987.

[LY90] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

[LZ91] S. Y. Le and M. Zuker. Predicting common foldings of homologous RNAs. *Journal of Biomolecular Structure and Dynamics*, 8:1027–1044, 1991.

[MAHK91] J. Merriam, M. Ashburner, D. L. Hartl, and F. C. Kafatos. Toward cloning and mapping the genome of *Drosophila*. *Science*, 254:221–225, 1991.

[Mar86] C. C. Marvel. A program for the identification of tRNA-like structure in DNA data. *Nucleic Acids Research*, pages 431–435, 1986.

[MECS90] F. Michel, A. D. Ellington, S. Couture, and J. W. Szostak. Phylogenetic and genetic evidence for base-triples in the catalytic domain of group I introns. *Nature*, 347:578–580, 1990.

[MEK66] J. T. Madison, G. A. Everett, and H. K. Kung. On the nucleotide sequence of yeast tyrosine transfer RNA. *Cold Spring Harbor Symposium on Quantitative Biology*, 31:409–416, 1966.

[Mia94] I. Saira Mian. Private e-mail communication, April 1994.

[Min93] Y. Minobe. Analysis of rice genome. *Tanpakushitsu Kakusan Koso. Protein, Nucleic Acid, Enzyme*, 38:704–712, 1993.

[MMCN93] T. Macke, I. S. Mian, P. Cohen, and H. F. Noller. stgrep: a language for performing symbolic computation on RNA sequences and secondary structure. In preparation, 1993.

[MUO89] F. Michel, K. Umesono, and H. Ozeki. Comparative and functional anatomy of group II catalytic introns–a review. *Gene*, 82:5–30, 1989.

[MW90] F. Michel and E. Westhof. Modelling of the three-dimensional architecture of group I catalytic introns based on comparative sequence analysis. *Journal of Molecular Biology*, 216:585–610, 1990.

[NKW$^+$91] H. F. Noller, J. Kop, V. Wheaton, J. Brosius, R. R. Gutell, A. M. Kopylov, F. Dohme, W. Herr, D. A. Stahl, R. Gupta, and C. R. Woese. Secondary structure model for 23S ribosomal RNA. *Nucleic Acids Research*, 9:6167–6189, 1991.

[NPGK78] R. Nussinov, G. Pieczenik, J. R. Griggs, and D. J. Kleitman. Algorithms for loop matchings. *SIAM Journal of Applied Mathematics*, 35:68–82, 1978.

[NW81] H. F. Noller and C. R. Woese. Secondary structure of 16S ribosomal RNA. *Science*, 212:403–411, 1981.

[Oga93] N. Ogasawara. Sequencing project of *Bacillus subtilis* genome. *Tanpakushitsu Kakusan Koso. Protein, Nucleic Acid, Enzyme*, 38:669–676, 1993.

[Ols93] M. V. Olson. The human genome project. *Proceedings of the National Academy of Sciences of the U.S.A.*, 90:4338–4344, 1993.

[OS93] K. Okada and Y. Shimura. Genome studies of *Arabidopsis thaliana. Tanpakushitsu Kakusan Koso. Protein, Nucleic Acid, Enzyme*, 38:713–719, 1993.

[OvAC$^+$92] S. G. Oliver, Q. J. van der Aart, M. L. Agostoni-Carbone, M. Aigle, L. Alberghina, D. Alexandraki, G. Antoine, R. Anwar, J. P. Ballesta, and P. Benit. The complete DNA sequence of yeast chromosome III. *Nature*, 357:38–46, 1992.

[OYO$^+$92] K. Oda, K. Yamato, E. Ohta, Y. Nakamura, M. Takemura, N. Nozato, T. Kohchi, Y. Ogura, T. Kanegae, K. Akashi, and K. Ohyama. Gene organization deduced from the complete sequence of liverwort *Marchantia polymorpha* mitochondrial DNA. *Journal of Molecular Biology*, 223:1–7, 1992.

[PBDB93] G. Plunkett 3d, V. Burland, D. L. Daniels, and F. R. Blattner. Analysis of the *Escherichia coli* genome. III. DNA sequence of the region from 87.2 to 89.2 minutes. *Nucleic Acids Research*, 21:3391–3398, 1993. E. coli sequencing project.

[PC93] S. R. Presnell and F. E. Cohen. Artificial neural networks for pattern recognition in biochemical sequences. *Annual Review of Biophysics and Biomolecular Structure*, 22:283–298, 1993.

[Ple90] C. W. A. Pleij. Pseudoknots: a new motif in the RNA game. *Trends in Biochemical Sciences*, 15:143–147, 1990.

[Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc IEEE*, 77(2):257–286, 1989.

[RB91] D. P. Romero and E. H. Blackburn. A conserved secondary structure for telomerase RNA. *Cell*, 67:343–353, 1991.

[RSF$^+$66] U. L. RajBhandary, A. Stuart, R. D. Faulkner, S. H. Chang, and H. Khorana. Nucleotide sequence studies of yeast phenylalanine sRNA. *Cold Spring Harbor Symposium on Quantitative Biology*, 31:425–434, 1966.

[SA90] P. R. Sibbald and P. Argos. Scrutineer: a computer program that flexibly seeks and describes motifs and profiles in protein sequence databases. *Computer Applications in the Biosciences*, 6(3):279–288, 1990.

[SAB$^+$77] F. Sanger, G. M. Air, B. G. Barrell, N. L. Brown, A. R. Coulson, C. A. Fiddes, C. A. Hutchison, P. M. Slocombe, and M. Smith. Nucleotide sequence of bacteriophage $\phi$X174 DNA. *Nature*, 265:687–695, 1977.

[Sae84] W. Saenger. *Principles of nucleic acid structure.* Springer Advanced Texts in Chemistry. Springer-Verlag, New York, 1984.

[Sak92] Y. Sakakibara. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97:23–60, 1992.

[San85] D. Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, 45:810–825, 1985.

[SBDC93] M. E. Schmitt, J. L. Bennett, D. J. Dairaghi, and D. A. Clayton. Secondary structure of RNase MRP RNA as predicted by phylogenetic comparison. *Faseb Journal*, 7:208–213, 1993.

[SBH$^+$93] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. Underwood, and D. Haussler. The application of stochastic context-free grammars to folding, aligning and modeling homologous RNA sequences. Technical Report UCSC-CRL-94-14, University of California, Santa Cruz, Computer and Information Sciences Dept., Santa Cruz, CA 95064, 1993.

[SBH$^+$94a] Yasubumi Sakakibara, Michael Brown, Richard Hughey, I. Saira Mian, Kimmen Sjölander, Rebecca C. Underwood, and David Haussler. Recent methods for RNA modeling using stochastic context-free grammars. In *Proceedings of the Asilomar Conference on Combinatorial Pattern Matching*, New York, NY, 1994. Springer-Verlag. In press.

[SBH$^+$94b] Yasubumi Sakakibara, Michael Brown, Richard Hughey, I. Saira Mian, Kimmen Sjölander, Rebecca C. Underwood, and David Haussler. Stochastic context-free grammars for tRNA modeling. Submitted to NAR, 1994. NAR SCFG paper.

[SBM$^+$94] Y. Sakakibara, M. Brown, I. S. Mian, R. Underwood, and D. Haussler. Stochastic context-free grammars for modeling RNA. In *Proceedings of the Hawaii International Conference on System Sciences*, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[SBU$^+$93] Y. Sakakibara, M. Brown, R. Underwood, I. S. Mian, and D. Haussler. Stochastic context-free grammars for modeling RNA. Technical Report UCSC-CRL-93-16, UC Santa Cruz, Computer and Information Sciences Dept., Santa Cruz, CA 95064, 1993.

[SCF$^+$78] F. Sanger, A. R. Coulson, T. Friedmann, G. M. Air, B. G. Barrell, N. L. Brown, J. C. Fiddes, C. A. Hutchison 3d, P. M. Slocombe, and M. Smith. The nucleotide sequence of bacteriophage $\phi$X174. *Journal of Molecular Biology*, 125:225–246, 1978.

[SCH$^+$82] F. Sanger, A. R. Coulson, G. F. Hong, D. F. Hill, and G. B. Petersen. Nucleotide sequence of bacteriophage lambda DNA. *Journal of Molecular Biology*, 162:729–773, 1982.

[SCZ$^+$80] R. Stiegler, P. Carbon, M. Zuker, J. P. Ebel, and C. Ehresmann. Secondary and topographic structure of ribosomal RNA 16S of *escherichia coli*. *Academy Sciences (Paris) Series D*, 291:937–940, 1980.

[SD89] T. J. Santner and D. E. Duffy. *The Statistical Analysis of Discrete Data*. Springer Verlag, New York, 1989.

[SD93] D. B. Searls and S. Dong. A syntactic pattern recognition system for DNA sequences. In *Proc. 2nd Int. Conf. on Bioinformatics, Supercomputing and complex genome analysis*. World Scientific, 1993. In press.

[SDT$^+$92] J. Sulston, Z. Du, K. Thomas, R. Wilson, L. Hillier, R. Staden, N. Halloran, Green P., J. Thierry-Mieg, L. Qiu, S. Dear, A. Coulson, M. Craxton, R. Durbin, M. Berks, M. Metzstein, T. Hawkins, R. Ainscough, and R. Waterston. The *C. elegans* genome sequencing project: a beginning. *Nature*, 356:37–41, 1992.

[Sea92] David B. Searls. The linguistics of DNA. *American Scientist*, 80:579–591, November–December 1992.

[Sea93a] D. B. Searls. The computational linguistics of biological sequences. In *Artificial Intelligence and Molecular Biology*, chapter 2, pages 47–120. AAAI Press, 1993.

[Sea93b] D. B. Searls. String variable grammar: a logic grammar formalism for DNA sequences, 1993. Unpublished.

[Sha88] B. A. Shapiro. An algorithm for comparing multiple RNA secondary structures. *CABIOS*, 4(3):387–393, 1988.

[SM87] W. Saurin and P. Marlière. Matching relational patterns in nucleic acid sequences. *Computer Applications in the Biosciences*, 3(2):115–120, 1987.

[Sta80] R. Staden. A computer program to search for tRNA genes. *Nucleic Acids Research*, pages 817–825, 1980.

[Sta90] R. Staden. Searching for patterns in protein and nucleic acid sequences. In Doolittle [Doo90], pages 193–211.

[STG92] D. Schneider, C. Tuerk, and L. Gold. Selection of high affinity RNA ligands to the bacteriophage R17 coat protein. *Journal of Molecular Biology*, 228:862–869, 1992.

[Sut79] J. G. Sutcliffe. Complete nucleotide sequence of the *Escherichia coli* plasmid pBR322. *Cold Spring Harbor Symposia on Quantitative Biology*, 43:77–90, 1979.

[SZ90] B. A. Shapiro and K. Zhang. Comparing multiple RNA secondary structures using tree comparisons. *CABIOS*, 6(4):309–318, 1990.

[TE93] A. J. Tranguch and D. R. Engelke. Comparative structural analysis of nuclear RNase P RNAs from yeast. *Journal of Biological Chemistry*, 268:14045–1455, 1993.

[THSH92] C. S. Tzeng, C. F. Hui, S. C. Shen, and P. C. Huang. The complete nucleotide sequence of the *Crossostoma lacustre* mitochondrial genome: conservation and variations among vertebrates. *Nucleic Acids Research*, 20:4853–4858, 1992.

[tPD92] E. ten Dam, K. Pleij, and D. Draper. Structural and functional aspects of RNA pseudoknots. *Biochemistry*, 31:11665–11676, 1992.

[TSF88] D. H. Turner, N. Sugimoto, and S. M. Freier. RNA structure prediction. *Annual Review of Biophysics and Biophysical Chemistry*, 17:167–192, 1988.

[TUL71] I. Tinoco Jr., O. C. Uhlenbeck, and M. D. Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230:363–367, 1971.

[TW68] J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2:57–81, 1968.

[vDF$^+$92] M. von Montagu, C. Dean, R. Flavell, H. Goodman, M. Koornneef, E. Meyerowitz, J. Peacock, Y. Shimura, and C. Somerville. The multinational coordinated *Arabidopsis thaliana* genome research project. Progress Report: year two. Technical Report NSF 90-80, National Science Foundation, Washington DC, 1992.

[WAG84] M. S. Waterman, R. Arratia, and D. J. Galas. Pattern recognition in several sequences: consensus and alignment. *Bulletin of Mathematical Biology*, 46:515–527, 1984.

[Wat88] M. S. Waterman. Computer analysis of nucleic acid sequences. *Methods in Enzymology*, 164:765–792, 1988.

[Wat89] M. S. Waterman. Consensus methods for folding single-stranded nucleic acids. In M. S. Waterman, editor, *Mathematical Methods for DNA Sequences*, chapter 8. CRC Press, 1989.

[WGGN83] C. R. Woese, R. R. Gutell, R. Gupta, and H. F. Noller. Detailed analysis of the higher-order structure of 16S-like ribosomal ribonucleic acids. *Microbiology Reviews*, 47(4):621–669, 1983.

[WGN93] B. Weiser, R. Gutell, and H. F. Noller. XRNA: An X Windows environment RNA editing/display package. Unpublished manuscript, January 1993.

[WMG+80] C. R. Woese, L. J. Magrum, R. Gupta, R. B. Siegel, D. A. Stahl, J. Kop, N. Crawford, J. Brosius, R. Gutell, J. J. Hogan, and H. F. Noller. Secondary structure model for bacterial 16S ribosomal RNA: phylogenetic, enzymatic and chemical evidence. *Nucleic Acids Research*, 8:2275–2293, 1980.

[WOW+90] S. Winker, R. Overbeek, C.R. Woese, G.J. Olsen, and N. Pfluger. Structure detection through automated covariance search. *CABIOS*, 6:365–371, 1990.

[WPT89] J. R. Wyatt, J. D. Puglisi, and I. Tinoco Jr. RNA folding: pseudoknots, loops and bulges. *BioEssays*, 11(4):100–106, 1989.

[ZDF+66] H. G. Zachau, D. Dütting, H. Feldmann, F. Melchers, and W. Karau. Serine specific transfer ribonucleic acids. XIV. Comparison of nucleotide sequences and secondary structure models. *Cold Spring Harbor Symposium on Quantitative Biology*, 31:417–424, 1966.

[ZGB81] C. Zwieb, C. Glotz, and R. Brimacombe. Secondary structure comparisons between small subunit ribosomal RNA molecules from six different species. *Nucleic Acids Research*, 9:3621–3640, 1981.

[ZS84] M. Zuker and D. Sankoff. RNA secondary structures and their prediction. *Bull. Math. Biol.*, 46:591–621, 1984.

[Zuk89a] M. Zuker. On finding all suboptimal foldings of an RNA molecule. *Science*, 244:48–52, 1989.

[Zuk89b] M. Zuker. On finding all suboptimal foldings of an RNA molecule. *Science*, 244:48–52, 1989.

[Zwi89] C. Zwieb. Structure and function of signal recognition particle RNA. *Progress in Nucleic Acid Research and Molecular Biology*, 37:207–234, 1989.