Algebraic and Combinatorial Properties of Common RNA Pseudoknot Classes with Applications

Markus E. Nebel*, Frank Weinberg
Department of Computer Science, University of Kaiserslautern, Germany

Abstract

Predicting RNA structures with pseudoknots in generell is an NP-complete problem. Accordingly, several authors have suggested subclasses which provide polynomial time prediction algorithms by allowing resp. disallowing certain structural motives. In this paper we introduce a unifying algebraic view on most of these classes. That way it becomes possible to find linear time recognition algorithms that decide whether or not a given structure is member of a class (we offer these algorithms as a web-service to the scientific community). Furthermore, by presenting a general translation scheme of our algebraic descriptions into multiple context-free grammars and proving a new correspondence of multiple context-free grammars and generating functions it becomes possible to derive the precise asymptotic size of all the classes solving some open problems like enumerating the Rivas & Eddy class of pseudoknots.

1 Introduction

To a large extent the function of an RNA molecule is determined by its secondary structure, i.e., by the mutual arrangements of the base paired helices. In this context however, secondary structure has to be understood in a wider sense by allowing pseudoknots. Even though the vast majority of RNAs has simple, i.e., pseudoknot-free, secondary structure, `PseudoBase` [27] lists more than 300 records of pseudoknots determined by a variety of experimental and computational techniques including crystallography, NMR, mutational experiments, and comparative sequence analysis. If present, pseudoknots in many cases are crucial for molecular function. Examples include the catalytic cores of several ribozymes [8], telomerase activity [28], reviewed in [26, 10], and programmed frameshifting [19]. Accordingly, one aims for *in silico* prediction of RNA structure which unfortunately has been proven $\mathcal{NP}$-complete when allowing unrestricted conformations even under the assumption of rather simple energy models [1, 16]. However, polynomial-time algorithms have been devised, for certain restricted classes of pseudoknots. The following references provide a certainly incomplete list of those approaches: [1, 3, 4, 7, 13, 15, 17, 20, 21, 22, 29]. The interrelationships of some of these classes have been partially clarified by [6] and [23]. Furthermore, [24] presents enumeration results quantifying the size of some of those classes. In addition to the before mentioned exact algorithms, plenty of heuristic approaches to pseudoknot prediction have been suggested; see e.g., [5, 18] and the references therein.

In this contribution, we provide a unified algebraic characterization of all the before mentioned classes connected to exact approaches. For this purpose we make use of *operators* like nesting or stem extension to show how each class can be understood as the smallest set of conformations closed under a properly chosen combination of operators. This way it is possible to understand the interrelations between the classes, proving inclusion or incomparability. Additionally, we are able to derive linear time recognition algorithms for all the classes. To the best of our knowledge those have not been available in all cases so far. Furthermore, we show how all the operators used can be translated into unambiguous multiple context-free grammars (MCFG). As a consequence grammar

*This research has been supported in part by DFG research grant NE 1379/3-1.

representations for all the pseudoknot classes are found. By proving a new enumeration lemma that allows to compute the number of different words of length n in the language generated by any unambiguous MCFG, we are able to provide exact asymptotics for the sizes of all the classes. This way we are able to provide the first results for the size of Rivas & Eddy's pseudoknot class solving a problem now open for more than 10 years.

2 Definitions of pseudoknot classes

The pseudoknot free (PKF) class

For completeness we also include the class of pseudoknot free secondary structures in our treatment. It is easily verified that any pseudoknot free secondary structure can be created by taking a number of base pairs and unpaired bases and concatenate or embed them into the growing structure. Note that we assume in the entire of this paper a minimal hairpin loop length of zero. This is no restriction implied by our ideas or methods but helps to reduce the complexity of the presentation.

The Rivas and Eddy (R&E) class

In [22] Rivas and Eddy introduced an extension of the traditional dynamic programming scheme of the Zuker algorithm [31] that allows to predict a huge class of pseudoknots.

The original algorithm of Zuker basically determines the optimal folding for any connected region of the RNA molecule by considering either to add a pairing consisting of the first and last base of the region to the optimal folding of the remaining region or to split it into two smaller connected regions for which the optimal folding has been determined earlier. As a base case regions of size 1 always consist of a single unpaired nucleotide.

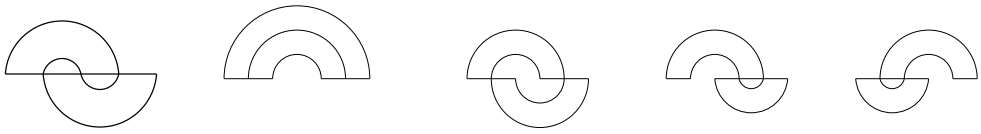

Figure 1: Decomposing a connected or gapped region into two gapped regions.

The new algorithm of Rivas and Eddy additionally considers regions that are not entirely connected but have a single gap inside. Figure 1 shows the different ways how a connected or gapped region may be decomposed into two gapped regions. Here the horizontal lines pictures (segments of) the backbone of the molecule and a semi-disc in the upper resp. lower half-plane is used to represent a gapped region where the inner half-circle determines the gap and the positions between the two circles may be paired (such that the corresponding bonds depicted as half-circles run in parallel to the bordering ones). Additionally the algorithm can decompose a gapped region into a gapped and a connected region in all 4 possible ways and it may decompose a gapped region into two separate connected regions. The base case for gapped regions are regions consiting only of two paired bases.

The algorithm works in $\mathcal{O}(n^6)$ time and $\mathcal{O}(n^4)$ space. While it is known that the class of pseudoknots that can be predicted by it (called R&E) is a superclass of any other class for which an exact prediction algorithm has been suggested thus far and [6] gives a linear time algorithm that decides if a given structure is in R&E, neither the size of the class nor a short description of it (e.g. as a formal language) has been determined so far.

The Uemura et al. (UMK) class

In [29] Uemura et al. suggest a set of rules for tree adjoined grammars that can be used to generate a language that models pseudoknots. In [17] Matsui et al. translate these rules into the setting of

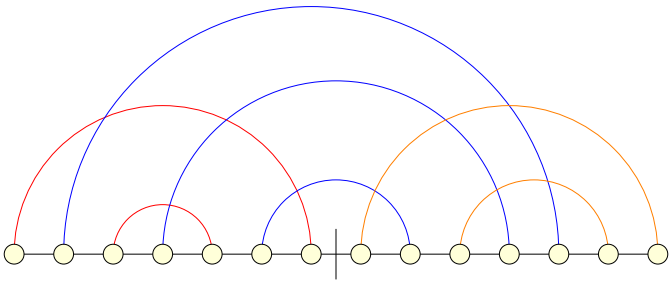

Figure 2: An UMK-pseudoknot.

pair stochastic tree adjoined grammars and in [13] Kato et al. give an equivalent set of rules for stochastic multiple context-free grammars. All algorithms run in $\mathcal{O}(n^5)$ time and $\mathcal{O}(n^4)$ space.

The construction of a secondary structure according to this set of rules can be viewed as starting with two insertion points and repeatedly inserting a base pair with one of its bases to the immediate left of one insertion point and the other base to the immediate right of either the same or the other insertion point. Additionally unpaired bases or substructures (of the same type) may be inserted at any point.

Thus a single UMK structure element consists of three groups of base pairs, each of which may be empty. Within each group all base pairs form a possibly interrupted *stem*, i.e. are nested as in the second picture of Figure 1, possibly with bases from other groups between them. The left and right group of base pairs do not overlap and there is a position between these groups such that all base pairs in the middle group span this position. See Figure 2 for an example.

An UMK structure then consists of any number of UMK structure elements arbitrarily concatenated and embedded inside each other.

For this class no previous results concerning the size or a linear time recognition algorithm or relation to other classes except R&E are known to the authors.

The Akutsu and Uemura (A&U) class

In [1] Akutsu proposes a minimum free energy based dynamic programming algorithm that predicts pseudoknots from a subclass of UMK in $\mathcal{O}(n^5)$ time.

The basic components of A&U structures are so-called simple pseudoknots. A simple pseudoknot consists of two groups of base pairs. Again within both groups the base pairs form a stem. Base pairs are arranged such that the right bases of the first group and the left bases of the second group are interleaved arbitrarily while the other bases all lie outside of the interleaved area. See Figure 3 for an example.

An A&U structure then consists of any number of pseudoknot free structures and simple pseudoknots arbitrarily concatenated and embedded inside each other.

The inclusion relation of A&U with most of the other classes has been examined in [6] and extended in [24], where additionally its size was determined. A linear time recognition algorithm

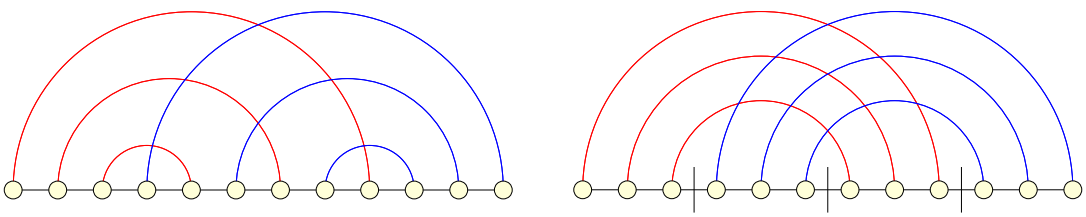

Figure 3: A simple pseudoknot (left) and a H-type pseudoknot (right).

however has not yet been given.

The Lyngsø and Pedersen (L&P, L&P$^+$) classes

In [15] Lyngsø and Pedersen present an algorithm based on the minimum free energy paradigma that runs in $\mathcal{O}(n^5)$ time and $\mathcal{O}(n^3)$ space and predicts structures of the form $w_1u_1w_2u_2w_3$, where $w_1w_2w_3$ and u_1u_2 are arbitrary pseudoknot free structures. This class of structures is labeled as L&P$^+$ in [6]. Additionally the authors of [6] also defined the subclass of structures $w_1u_1w_2u_2$, w_1w_2 and u_1u_2 pseudoknot free, and labeled that class as L&P.

For L&P [6] gives a linear time recognition algorithm along with some inclusion results and [24] extends the inclusion results and gives the size of the class. L&P$^+$ has not yet been further examined.

The Dirks and Pierce (D&P) class

In [7] Dirks and Pierce introduce a dynamic programming algorithm that runs in $\mathcal{O}(n^5)$ time and $\mathcal{O}(n^4)$ space.

The class of structures predicted by the D&P algorithm is similar to the UMK and A&U classes in that it allows concatenation and embedding at arbitrary positions. The base class of pseudoknots for this algorithm are so-called H-type pseudoknots.

H-type pseudoknots consist of two groups of base pairs, each nested as the groups in simple pseudoknots. The groups are arranged such that each base pair in the first group overlaps each base pair in the second group. See Figure 3 for an example.

For this class the size, a linear time recognition algorithm and many inclusion relations have been determined in [6] and [24].

The Reeder and Giegerich (R&G) Class

In [20] Reeder and Giegerich present an algorithm that predicts a class of H-type pseudoknots in $\mathcal{O}(n^4)$ time and $\mathcal{O}(n^2)$ space.

The R&G class uses the same set of base structures as D&P but restricts the embedding of unpaired bases and other structures to the positions indicated by vertical lines in Figure 3.

Size and inclusion results for this class have been determined in [24].

The Cao and Chen (C&C) classes

In [3] Cao and Chen give another algorithm for predicting H-type pseudoknots. This algorithm is based on a more sophisticated energy model than the previous ones and takes $\mathcal{O}(n^6)$ time.

The class of structures covered by this algorithm is very similar to R&G in that embedding substructures is restricted to the positions marked in Figure 3. However unpaired bases may be embedded anywhere.

It is not completely clear from [3] if embedding substructures at the middle one of the marked positions is allowed. The authors of [24] defined the class where this is forbidden as C&C. We decided to also include the other case and named the resulting class C&C$^+$.

Size and inclusion results for both classes are given in [24].[1]

The Chen, Condon and Jabbari (CCJ) class

In [4] Chen, Condon and Jabbari introduce a dynamic programming algorithm that takes $\mathcal{O}(n^5)$ time and $\mathcal{O}(n^4)$ space. Like the other algorithms with that time and space complexity it predicts a class of pseudoknots that results from concatenating and embedding at arbitrary positions structures from a base class.

Here the base class consists of pseudoknot free structures and CCJ pseudoknots. A CCJ pseudoknot in turn consists of two TGB gapped structures that are combined as in the first

[1] In [24] unpaired bases are ignored making C&C$^+$ identical to R&G in their setting.

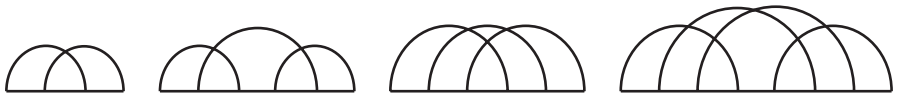

Figure 4: All shadows with topological genus 1.

picture of Figure 1. A TGB gapped structure finally is a UMK structure element with a gap inserted at the position marked by a vertical bar in Figure 2.

For this class some inclusion results have been given in [4] but neither a recognition algorithm nor the size of the class are known.

The Reidys and Nebel (R&N) class

The approach of Reidys et al. in [21] differs from most of the other ones in that they first define a class of structures they want to be able to predict and then develop an algorithm to do this instead of giving an algorithm and defining the class (implicitly) as the set of structures predicted by this algorithm.

The authors categorize secondary structures by the topological genus of their representation as a graph and show that the pseudoknot free structures are exactly the ones with genus 0. Furthermore they show that pseudoknots with genus 1 are exactly the ones that have one of the shadows listed in Figure 4, where the *shadow* of a pseudoknot is the structure that results from deleting all unpaired positions and reducing all uninterrupted stems to a single base pair.

Finally an R&N structure consists of any number of pseudoknot free structures and genus 1 pseudoknots arbitrarily concatenated and embedded inside each other.

For this class some inclusion results are presented in [21] but neither the size of the class nor an linear time recognition algorithm have been presented yet.

3 Algebraic Classification

In Section 2 we introduced several classes of pseudoknotted structures. In this section we will show that these classes can be created from a single arc (base pair) by using just three basic operations.

We will denote the classes as languages over $\Sigma = \{[_i,]_i, \bullet \mid i \in \mathbb{N}\}$ with each pair $[_i]_i$ appearing at most once in each word.

We will call two words w_1 and w_2 equivalent (notation $w_1 \equiv w_2$) if they can be transformed into each other by renumbering the pairs of parentheses and introduce the convention for all languages L that if $w \in L$ and $w' \equiv w$ then $w' \in L$ as an implicit extension of the definitions below.

We will call w_1 and w_2 disjoint ($w_1 \perp w_2$) if no pair $[_i]_i$ of parentheses appears in both of them. In this notation the operations we are going to use are as follows:

Definition 3.1. *Let $C, C' \subset \Sigma^*$. We call $B \subset \Sigma^*$ resulting from C (resp. C and C') by*

- nesting *if* $B = n'(C, C') := C \cup \{w_1 u w_2 \mid w = w_1 w_2 \in C, u \in C', u \perp w\}$
  or $B = n(C) := n'(C, C)$
  or $B = n''(C) := C \cup \{w_1 u w_2 \mid w = w_1 w_2, u \in C, u \perp w$ *and*
  if $(w_1, w_2) = (w_1'[_i,]_j w_2')$ *then* $i = j\}$.

- stem extension *if* $B = s(C) :=$
  $C \cup \{w_1[_j[_i w_2]_i]_j w_3, w_1[_i[_j w_2]_j]_i w_3 \mid w = w_1[_i w_2]_i w_3 \in C, w \perp [_j]_j\}$.

- knot extension *if* $B = k(C) :=$
  $C \cup \{w_1[_j[_i]_j w_2]_i w_3, w_1[_i w_2[_j]_i]_j w_3, w_1[_i[_j w_2]_i]_j w_3 \mid w = w_1[_i w_2]_i w_3 \in C, w \perp [_j]_j\}$
  or $B = k_l'(C) :=$
  $C \cup \{w_1[_j[_i]_j w_2]_i w_3, w_1[_i w_2[_j]_i]_j w_3 \mid w = w_1[_i w_2]_i w_3 \in C, i \leq l, w \perp [_j]_j\}$.

Note that in each of the cases any of the components w_i may be empty.

Since all operations work by adding new words to C we can denote the closure of C under $f \in \{n, n'', s, k, k'\}$ by $f^\infty(C)$ and the closure under two operations f and g by $(f \circ g)^\infty(C)$. In the case of n' we define the closure $n'^\infty(C, C') := n'(n'(\ldots n'(C, C'), \ldots, C'), C')$.

The set A of basic structures we will use is $A = \{\epsilon, [_1]_1\}$ if we consider structures without unpaired bases or $A = \{\epsilon, [_1]_1, \bullet\}$ if we include unpaired bases.

In order to translate the descriptions from Section 2 into algebraic specifications we make use of the following observations:

- The nesting operation $n(C)$ covers concatenation as well as embedding at arbitrary positions. Thus $n^\infty(B)$ describes the class that results from arbitrarily concatenating and embedding into each other any number of structures from B.
- For a class C of shadows, $s^\infty(C)$ describes exactly the pseudoknots with shadows in C.
- The restricted embedding of the R&G, C&C and C&C$^+$ classes can be achieved by embedding freely into the shadows of the pseudoknots. The additional restriction of C&C is modeled by n''.
- $k(A)$ is A extended by the shadow of H-type pseudoknots $[_1[_2]_1]_2$. We will use it like a base class in the following.
- The structures from L&P can be described as a single H-type pseudoknot with embedded or concatenated pseudoknot free structures or completely pseudoknot free structures. For L&P$^+$ the single pseudoknot can additionally have one of the shadows $[_1[_2[_3]_2]_3]_1$, $[_1[_2]_1[_3]_2]_3$ or $[_1[_2[_3]_2[_4]_3]_4]_1$.

Using these observations and defining the base classes $U = A \setminus \{[_1]_1\}$, $L = k(A) \cup \{[_1[_2[_3]_2]_3]_1$, $[_1[_2]_1[_3]_2]_3$, $[_1[_2[_3]_2[_4]_3]_4]_1\}$, $Si = A \cup \{s \mid s$ is shadow of a simple pseudoknot$\}$ and $G = A \cup \{s \mid s$ is shadow of a genus-1-pseudoknot$\}$ we find:

$$\begin{array}{lcl}
\text{PKF} & = & n^\infty(A), \\
\text{L\&P} & = & n'^\infty(s^\infty(k(A)), A), \\
\text{L\&P}^+ & = & n'^\infty(s^\infty(L), A), \\
\text{R\&G} & = & s^\infty(n^\infty(k(A))), \\
\text{C\&C} & = & n'^\infty(s^\infty(n''^\infty(k(A))), U), \\
\text{C\&C}^+ & = & n'^\infty(s^\infty(n^\infty(k(A))), U), \\
\text{D\&P} & = & (s \circ n)^\infty(k(A)), \\
\text{A\&U} & = & (s \circ n)^\infty(Si), \\
\text{R\&N} & = & (s \circ n)^\infty(G), \\
\text{UMK} & = & (s \circ n \circ k'_1)^\infty(A), \\
\text{CCJ} & = & (s \circ n \circ k'_2)^\infty(k(A)), \\
\text{R\&E} & = & (s \circ n \circ k)^\infty(A).
\end{array}$$

From these descriptions we can easily conclude the inclusion that the hierarchy of the classes is as depicted in Figure 5:

Lemma 3.1.

- *PKF* $\subset$ *R&G* $\subset$ *C&C*$^+$ $\subset$ *D&P* $\subset$ *A&U* $\subset$ *UMK* $\subset$ *CCJ* $\subset$ *R&E,*
- *PKF* $\subset$ *L&P* $\subset$ *L&P*$^+$ $\subset$ *R&N* $\subset$ *R&E,*
- *PKF* $\subset$ *C&C* $\subset$ *C&C*$^+$*,*
- *L&P* $\subset$ *D&P* $\subset$ *R&N,*
- *L&P*$^+$ $\subset$ *UMK.*

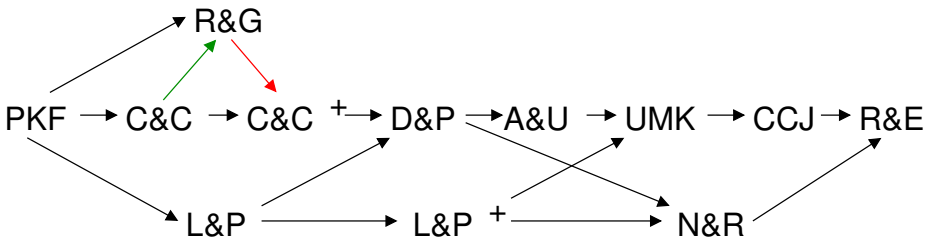

Figure 5: Hierarchy of pseudoknot classes.

- *C&C ⊂ R&G if and only if unpaired bases are left out.*
- *Without unpaired bases R&G = C&C, all other inclusions are proper.*
- *Pairs of classes not covered above are set-theoretically incomparable.*

Proof. Since A contains only one type of parentheses we find $n^{\infty}(A) = n''^{\infty}(A)$ giving PKF ⊂ C&C.

The elements of Si can be generated from A by alternatingly applying the second case of knot extension and stem extension, each time with $[_1]_1$ as the basis, starting and ending with a knot extension. Thus $Si \subset (s \circ k'_1)^{\infty}(A)$ giving A&U ⊂ UMK.

Since each element of G can be created by applying k at most three times to A we find $G \subset k^3(A)$. Additionally $k(A) \subset L \subset n(G)$, thus D&P ⊂ R&N ⊂ R&E and L&P$^+$ ⊂ R&N ∩ UMK.

The other inclusions are immediate.

For the incomparability results note that

- $[_1[_2[_3]_2]_3[_4]_1]_4 \in$ (C&C∩R&G)\L&P$^+$,
- $[_1[_2]_2[_3[_4]_3]_1]_4 \in$ L&P\C&C$^+$,
- $[_1[_2]_1[_3]_2]_3 \in$ L&P$^+$ \A&U,
- $[_1[_2[_3]_2[_4]_1]_4]_2 \in$ A&U\R&N and
- $[_1\bullet[_2[_3]_2]_1]_3 \in$ C&C\R&G.

Finally, the fact that the inclusions are proper follows from the size results listed in Table 3. □

3.1 Recognition algorithm

The algebraic descriptions of the previous section can also be used to generate membership tests for the classes: A word w is in a class C, if and only if w can be reduced to a word from the base class B by repeatedly applying the inverse of the operations that build C from B. This gives Algorithm 1.

Theorem 3.2. *For C any of the classes defined in Section 2 and any word w, Algorithm 1 returns true, if and only if $w \in C$.*

Proof. We will break down the proof into three claims. Claim 1 will show that for each $w \in C$ there is a reduction using the rules of the algorithm. Claim 2 will show that if $w \in C$ and at some point multiple reduction steps are possible, each will lead to a complete reduction. Finally, Claim 3 will show that the algorithm will not miss any possible reductions, completing the proof of the "if"-part. Claim 4 will then show the "only-if"-part by proving that a word not in C can be reduced to a word in C by the algorithm only if at the same time the reduction is inhibited by the values of the *type*-flags.

Algorithm 1 Generic recognition algorithm. The same subscript indicates bases paired with each other; the base currently marked by the pointer p is indicated by an additional subscript p or, if its type doesn't matter, it is represented by p as a symbol of the word.

Input: Word w, algebraic description of class C
Output: true, if $w \in C$, false, if not

for each Closure Cl in the construction of C from outer to inner **do**
  for all base pairs i in w **do** $type[i] := none$ **end for**;
  $p := w.first$;
  while p is inside w **do**
    if Cl contains $s(B)$, $w = w_1[_i[_j w_2]_j]_{i,p} w_3$, $type[i], type[j] \neq restricted$ and
    either $type[i] = none$ or $type[j] = none$ **then**
      $w := w_1[_i w_2]_{i,p} w_3$;
      if $(type[j] \neq none)$ **then** $type[i] := base$ **end if**;
    else if Cl contains $k(B)$ and $w = w_1[_j[_i]_j w_2]_{i,p} w_3$
    or $w = w_1[_i w_2[_j]_i]_{j,p} w_3$ or $w = w_1[_i[_j w_2]_i]_{j,p} w_3$ **then**
      $w := w_1[_i w_2]_{i,p} w_3$;
    else if Cl contains $k'_m(B)$, $w = w_1[_j[_i]_j w_2]_{i,p} w_3$, $type[i] \neq restricted$
    and $type[j] \in \{none, simple\}$ **then**
      $w := w_1[_i w_2]_{i,p} w_3$;
      if $(type[j] = simple)$ **then** $type[i] := restricted$**;**
      else $type[i] := base$**;**
      end if
    else if Cl contains $k'_m(B)$, $w = w_1[_i w_2[_j]_i]_{j,p} w_3$ and $type[j] = none$ **then**
      $w := w_1[_i w_2]_{i,p} w_3$;
      if $(type[i] = none)$ **then** $type[i] := simple$;
      end if
    else if Cl contains $n(Si)$ and $w = w_1[_i[_k w_2[_l]_k[_j]_i]_j]_{l,p} w_3$ **then**
      $w := w_1[_k w_2[_l]_k]_{l,p} w_3$;
    else if Cl contains $n(B), n'(B', B)$ or $n''(B)$, $w = w_1 w_2 p w_3$ and $w_2 p \in B$ **then**
      if either the nesting is $n''(B)$, $w_1 = w'_1[_i$ and $w_3 =]_j w'_3$, $j \neq i$
      or $w_2 p = [_i[_j]_i]_j$ and $type[i]$ or $type[j]$ is $restricted$ **then**
        $p := p.right$;
      else
        $w := w_1 w_3$; $p := w_3.first$;
      end if
    else
      $p := p.right$;
    end if
  end while
end for
if w is in the base class of the innermost closure **then**
  return true;
else
  return false;
end if

Claim 1: Each construction possible in the specifications from Section 3 can be inverted by the reductions in Algorithm 1.

The only operation that is not covered by an immediate inverse is the nesting of structures not in the base class. However since nesting of non-base structures only occurs as part of the innermost closure, the resulting structures can also be created by extending the base structure after nesting instead of before, giving a construction that can be inverted.

Claim 2: If $w \in C$ and w' results from w by removing any $\bullet$ or corresponding pair of parentheses $[_i,]_i$ then $w' \in C$.

Take any construction of w according to the algebraic specification of C. If during this construction $[_i,]_i$ is used as the basis for a stem extension or a knot extension of the third kind, remove the last of these steps. Otherwise if $[_i,]_i$ is itself introduced during a stem or knot extension, remove this step, if it is introduced by nesting v, nest the structure v' instead that results by removing $[_i,]_i$ from v. (If v is in a base class B then v' is also in B[2], otherwise this modification can be applied recursively to the construction of v.) Finally, if $[_i,]_i$ is used as the basis for knot extensions of the first or second kind after the changed/removed step, replace these by nesting $[_j]_j$, which is in all of the base classes. (Note that in all classes defined here knot extension only appears in closures that also contain nesting.) The result of these modifications is a construction of w' according to the algebraic specification of C.

Claim 3: If at the end of an iteration of the while-loop $w = w_1 p w_2$ then placing p anywhere in w_1 will not lead to any reduction rule being applicable.

We show this by induction on the number k of iterations of the while-loop during the current closure.

If $k = 0$ then $w_1 = \epsilon$ and the claim is trivially true.

Now assume the $k+1$-th iteration transforms $w' = w'_1 p w'_2$ into $w = w_1 p w_2$. By the induction hypothesis reduction rules can be applicable in w_1 only at bases that were either not present in w'_1 or are the rightmost base of a group of base pairs that have changed neighborhood status. We distinguish by the branch of the **if** that is taken in this iteration:

If reduction according to stem extension or knot extension has been performed, only the neighborhood of groups of base pairs including $[_i]_i$ has changed, but $]_i = p$ is not in w_1 and thus the rightmost base of the group can not be in w_1.

Analogously, if a nested structure has been removed, no neighborhood relations of groups of base pairs that lie completely within w_1 have changed.

If a nested candidate for a simple pseudoknot has been reduced, only the neighborhood of groups of base pairs including $[_k]_k$ has changed. However $[_k]_k$ can not be part of a stem extension since $]_k$ is surrounded by $[_l]_l$ and if $[_k]_k$ is part of a simple pseudoknot then the next reduction step on this pseudoknot also has to involve $]_l$ which is not in w_1.

If no reduction has been performed, only the last symbol of w_1 has been added, for which the algorithm has just verified the non-applicability of all reduction rules.

Claim 4: If Algorithm 1 transforms w' into w'' in any single step and $w'' \in C$ then either $w' \in C$ or all constructions of w'' according to the specification of C contain a base pair $[_i]_i$ that satisfies at least one of the following conditions:

- $type[i] \neq none$ in w'' and $[_i]_i$ is introduced through something else than nesting a base structure,
- $type[i] = restricted$ in w'' and $[_i]_i$ is introduced through something else than nesting a single base pair or is used as a basis for a knot extension of the first kind or a stem extension.

If $w' = w_1[_i[_k w_2[_l]_k[_j]_i]_j]_{l,p} w_3$ and $w'' = w_1[_k w_2[_l]_k]_{l,p} w_3$ then $w'_1[_k w_2[_l]_k]_l w'_3$, where w'_1 and w'_3 are subsequences of w_1 resp. w_3, is a simple pseudoknot only if $w'_1 = \epsilon$, w'_3 contains only closing parentheses and $w_2 = w'_2 w''_2$, where w'_2 contains only opening parentheses and $w''_2 = \mathrm{corr}(w'_1) \sqcup\!\sqcup \mathrm{corr}(w'_3)$ ($\sqcup\!\sqcup$ the shuffle of two words) and $\mathrm{corr}(v)$ gives the corresponding closing (opening) parentheses to the parentheses in v in reverse order. In this case $w'_1[_i[_k w_2[_l]_k[_j]_i]_j]_{l,p} w'_3$ is also a simple pseudoknot and the first case of the claim holds.

[2] If $B = Si$, v' is not necessarily in Si, but in $(s \circ n)(Si)$, which is also sufficient.

If $w' = w_1[_j[_i]_j w_2]_{i,p} w_3$ or $w' = w_1[_i w_2[_j]_i]_{j,p} w_3$, $w'' = w_1[_i w_2]_{i,p} w_3$ and $type[j] = none$ then either $[_i]_i$ can't be introduced through nesting a base structure or the inverse operation may be applied to w''.

If $w' = w_1[_j[_i]_j w_2]_{i,p} w_3$ and $type[j] = simple$ then all reductions applied to $[_j]_j$ during the run of Algorithm 1 must have been stem extensions of the second kind or knot extensions of the second kind. If $[_i]_i$ has been introduced through nesting a single base pair and has only been used as basis for knot extensions of the second kind, the reductions applied to $[_j]_j$ can be inverted by applying knot extensions of the first kind and stem extensions of the second kind to $[_i]_i$.[3]

For all of the other reductions the inverse operations may be applied freely in the respective closure when constructing the class. Thus the first case of the claim holds in these cases. □

Since the special rule to reduce simple pseudoknots covers all elements of $Si \setminus k(A)$, each of the conditions inside the while-loop only needs to check a constant number of relations between bases, their paired counterparts and their neighbors. Thus by representing w as a doubly linked list with additional links between paired bases (cp. [6]) each iteration can be done in $\mathcal{O}(1)$ and the complete algorithm will have linear runtime in the length of w. Our algorithms as a web-service can be used at `http://wwwagak.cs.uni-kl.de/KnotFilter`.

3.2 Coverage of Biological Structures

We used Algorithm 1 to test the various classes for their coverage of biological structures in PseudoBase (PBase) [27] as well as 16S and 23S rRNA and Group I and II Introns from the Gutell Database [2]. The results are listed in Table 1.

Class	PBase	16S	23S	Grp I Intron	Grp II Intron	Overall [%]
PKF	0	3	24	0	32	4.6
L&P	295	91	24	132	32	44.8
L&P$^+$	302	91	42	132	32	46.8
R&G	205	92	24	4	32	27.9
C&C	269	133	24	125	32	45.5
C&C$^+$	283	135	24	132	32	47.3
D&P	296	636	27	132	32	87.7
A&U	296	636	27	132	32	87.7
R&N	303	636	45	133	32	89.8
UMK	303	636	47	133	32	89.9
CCJ	303	636	47	133	32	89.9
R&E	304	637	57	133	32	90.9
Total	304	637	168	133	38	100

Table 1: Coverage of biological structures.

Since a number of the database entries contains isolated base pairs, i.e. base pairs which are not part of a stem, and these base pairs are sometimes considered tertiary instead of secondary structure, we did an additional run with these base pairs replaced by unpaired bases. The results of this run are listed in Table 2.

4 Grammars

It is an easy exercise to use Ogden's Lemma [11] to prove that the language describing any class of pseudoknotted structures is not context-free. As a consequence, in order to enumerate classes

[3] In the symmetric case the order of the checks ensures that the reductions will be done with $[_i]_i$ as basis. Thus no special treatment is necessary there.

Class	PBase	16S	23S	Grp I Intron	Grp II Intron	Overall [%]
PKF	0	3	42	0	32	6.0
L&P	295	92	42	132	32	46.3
L&P$^+$	302	92	42	132	32	46.9
R&G	224	92	42	118	32	39.7
C&C	269	135	42	125	32	47.1
C&C$^+$	283	137	42	132	32	48.9
D&P	296	637	50	132	32	89.6
A&U	296	637	50	132	32	89.6
R&N	303	637	168	133	32	99.5
UMK	303	637	168	133	32	99.5
CCJ	303	637	168	133	32	99.5
R&E	304	637	168	133	32	99.5
Total	304	637	168	133	38	100

Table 2: Coverage of biological structures with isolated base pairs removed.

of pseudoknotted structures, we need to change to more expressive grammars. An appropriate choice are the so-called *multiple context-free grammars* where intermediate symbols are allowed to have several components.

Definition 4.1 ([25],[30])**.** *A multiple context-free grammar (*MCFG*) is given by a tuple* $G = (I, d, T, P, S)$ *comprised of*

1. I*, a finite set of nonterminal symbols,*
2. d*, a function from* I *to* $\mathbb{N}$*, assigning a dimension* $d(A)$ *to each* $A \in I$*,*
3. T*, a finite set of terminal symbols,*
4. P*, a finite set of rules and*
5. *the* axiom $S \in I$ *with* $d(S) = 1$*.*

In MCFGs, we can derive from a nonterminal symbol $A \in I$ tuples or vectors of words whose dimension is given by $d(A)$. For a non-terminal $A \in I$ and an index $i \in \{1, \ldots, d(A)\}$, we write A_i for the ith component of A. Restricting $d(A) = 1$ for all non-terminals A yields a plain CFG. The general form of a multi-dimensional rule is again quite similar to CFGs, just that it is in terms of "vectors" instead of "scalars":

$$\begin{pmatrix} A_1 \\ \vdots \\ A_{d(A)} \end{pmatrix} \to \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_{d(A)} \end{pmatrix}, \alpha_k \in (I \cup C_G)^\star, C_G := \{A_i \mid A \in I \wedge 1 \leq i \leq d(A)\}, 1 \leq k \leq d(A).$$

In some cases it will be convenient to denote the premise resp. conclusion of such a rule by $\vec{A}$ or A resp. $\vec{\alpha}$ or α if there is no risk of confusions. We call C_G the set of *intermediate components*. If more than one instance of the multidimensional non-terminal $\vec{A}$ is used on the right-hand side of a rule we will denote their components by $A_i^{(1)}$, $A_i^{(2)}$, ... to indicate which components belong together.

Although the definition is quite simple, stressing some aspects might improve understanding:

- We do not restrict the order of intermediate components in the α_i. Accordingly, B_2 might jump in front of B_1. Actually, this is a vital feature of MCFGs.
- The α_i may contain arbitrary terminal strings, surrounding and separating the intermediate components.

However, the definition allows one component of an intermediate symbol to be used even if not all the other components (of that very same intermediate symbol) appear. These kinds of *deletions* are not wanted and in fact are not necessary (see Lemma 2.2 of [25]). Therefore, we will in the following enforce the additional restriction that each intermediate symbol appears in the conclusion of a rule completely or not at all – i.e. the same number of each of its components, labeled with the same superscripts.

The leftmost derivation relation $\Rightarrow_{lm} \subseteq (T \cup \mathcal{I})^\star \times (T \cup \mathcal{I})^\star$ for MCFGs is then defiend by

$$xA_{i_1}\beta_1 A_{i_2}\beta_2 \cdots A_{i_{d(A)}}\beta_{d(A)} \Rightarrow_{lm} x\alpha_{i_1}\beta_1\alpha_{i_2}\beta_2 \cdots \alpha_{i_{d(A)}}\beta_{d(A)}$$

iff $x \in T^\star$, $\vec{A} \to \vec{\alpha} \in P$, $\{i_1, i_2, \ldots, i_{d(A)}\} = \{1, 2, \ldots, d(A)\}$ and the components $A_{i_1}, A_{i_2}, \ldots, A_{i_{d(A)}}$ were introduced in the same derivation step as belonging together. So, leftmost derivation in MCFGs means expanding the intermediate whom the leftmost component belongs to. Since components may appear shuffled on the left, we have to re-index them.

Note that we do not need to introduce a concept of languages of tuples of strings. This is due to the restriction that the axiom S is always one-dimensional. As a consequence, the only way to introduce higher dimensions for the intermediate symbols (and rules) is to concatenate intermediate components within the conclusion of a one-dimensional rule (with premise S or any intermediate symbol *reachable* by S). Thus all the sentential-forms generated by any MCFG are one-dimensional with mixed occurrences of terminal symbols and intermediate components.

Again we define $\Rightarrow^\star_{lm}$ to be the reflexive and transitive closure of $\Rightarrow_{lm}$ and are finally able to define the language of a MCFG G:

$$\mathcal{L}(G) := \{w \in T^\star \mid S \Rightarrow_{lm} w\}.$$

4.1 Translating algebraic specifications into MCFGs

Now we will show how the specifications from Section 3 can be transformed into unambiguous MCFGs.

In order to be able to resemble the way our operations extend the words, we need to represent the symbols by nonterminals during construction. We will represent each pair of parentheses as a two-dimensional nonterminal like $\binom{Z_1}{Z_2}$ and each bullets by a one-dimensional nonterminal like Y.

In order to make the grammars unambiguous we need to disallow some rule applications, depending on which rule introduced a certain pair of parentheses (see below for details). To allow us to model this we will use different nonterminals to distinguish which operation introduced or acted on a specific pair of parentheses.

From this representation the terminal words can easily be derived by adding rules $\binom{Z_1}{Z_2} \to \binom{[}{]}$ and $Y \to \bullet$ for all applicable symbols. Note that to avoid the necessity of an infinite alphabet we abstract from the indices at this point and encode the information which bases are paired with each other only in the derivation tree. This is acceptable since all applications of unambiguous grammars we present or suggest actually rely on the 1-to-1 correspondence between derivation trees and structures which is still given.

The base sets A, U, $k(A)$, L and G can be created explicitly from the start symbol of the grammar. In the case of A we have $S \to \epsilon$, $S \to U$ and $S \to A_1A_2$, for $k(A)$ we add the rule $S \to K_1^{(1)}K_1^{(2)}K_2^{(1)}K_2^{(2)}$. Equivalently L and G can be generated by adding a rule for each structure in the class (see the L&P$^+$ and R&N-grammar below).

To generate the structures in Si we use the construction described in the proof of Lemma 3.1:

$$\binom{A_1}{A_2} \to \binom{M_1}{K_1M_2K_2}, \quad \binom{M_1}{M_2} \to \binom{N_1K_1}{K_2N_2}, \quad \binom{N_1}{N_2} \to \binom{M_1}{K_1M_2K_2},$$

with no other rules to replace $\vec{N}$, while $\vec{K}$ and $\vec{M}$ are eligible for further operations according to the specification of the class.

The operations can be translated into grammar rules as follows (in each of the rules $\vec{Z}$ resp. Y serves as a placeholder for any symbol eligible for the operation):

- **Nesting:** First we note that nesting two structures u and then v consecutively at the same position of w is equivalent to nesting the concatenated structure $v \cdot u$ at this position. Thus closure under the nesting operation can be achieved by nesting at most once at each position.

  Since not nesting a substructure at a given position is equivalent to nesting the empty word at that position and each of our classes contains the empty word we may simply nest exactly once at each eligible position.

  Furthermore nesting a structure u into $w = w_1 w_2$ with $w_1 = \epsilon$ is equivalent to concatenating u and w and can equally be achieved by nesting w into u with $u_2 = \epsilon$.

  Thus the closure $n^\infty(C)$ can be achieved by inserting the start symbol S to the immediate right of each symbol already present for each structure in C. If nesting is the first closure applied to a base set created by explicit rules like A, $k(A)$ or G, we can simply insert the start symbol at each eligible position into these rules, if nesting is the outermost closure we can replace the rules that insert the terminal symbols by $\binom{Z_1}{Z_2} \to \binom{[S}{]S}$ resp. $Y \to \bullet S$.

  Finally note that for any class C $(s \circ n)^\infty(C) = n^\infty(s^\infty(C))$ and $(s \circ n \circ k)^\infty(C) = n^\infty((s \circ k)^\infty(C))$.

  To create $n'^\infty(C, C')$ we use the same constructions, replacing S by the start symbol T of a subgrammar that creates C'. Additionally we have to add T to the left of structures that are in C but not in C', since in these cases reversing the concatenation does not work.

  Furthermore if $n'(C, C')$ is the outer of two closures and C' is a subset of the class C'' nested in the inner closure, we may only add T at positions that have not been eligible for nesting S earlier.

  The case that C' and C'' have nonempty intersection but C' is not a subset of C'' does not occur for the classes considered in this paper.

  Since we use $n''^\infty(C)$ only as a first closure we can easily identify the position, where nothing may be inserted and not place S at this position.

- **Stem extension:** A single stem extension can be achieved using $\binom{Z_1}{Z_2} \to \binom{O_1 I_1}{I_2 O_2}$.

  Since applying stem extension to $\vec{I}$ will create the same structure as applying it to $\vec{O}$, we disallow the former to avoid ambiguities.

  Additionally if Z_1 and Z_2 are immediately neighbored, the resulting structure will be equivalent to nesting $A_1 A_2$ between Z_1 and Z_2. Thus we have to disallow $\vec{A}$ from stem extension. For any of the other symbols we are guaranteed that their components can not occur immediately neighbored.

  For symbols $\vec{Z}$ that are eligible for modified knot extension k'_i we have to distinguish, which of the resulting symbols inherits this property, requiring two rules $\binom{Z_1}{Z_2} \to \binom{O_1 J_1}{J_2 O_2}$ and $\binom{Z_1}{Z_2} \to \binom{Q_1 I_1}{I_2 Q_2}$, where $\vec{J}$ and $\vec{Q}$ are the symbols eligible for further knot extensions and to avoid disambiguities $\vec{J}$ is disallowed from further stem extensions like $\vec{I}$ and it is required to be extended at least once more.

- **Knot extension:** The three cases of knot extension are modeled by the rules $\binom{Z_1}{Z_2} \to \binom{C_1 B_1 C_2}{B_2}$, $\binom{Z_1}{Z_2} \to \binom{D_1}{E_1 D_2 E_2}$ and $\binom{Z_1}{Z_2} \to \binom{L_1 R_1}{L_2 R_2}$. Again we have to disallow the symbols on the right-hand side from some rules to ensure that our grammars are unambiguous:

  The structures created by applying both the first and second type of knot extension are independent of the order of application. Thus we require all applications of the first type to happen before the second type is applied to the same arc by excluding D from knot extension of the first type.

  Applying stem extension to C (E) is equivalent to applying knot extension of the first (second) kind to B (D). Thus the first version is forbidden.

Since applying the third kind of knot extension to $\vec{L}$ will create the same structure as applying it to $\vec{R}$, we disallow the former.

In case of the modified knot extension k'_l the third rule is omitted and we have to additionally disallow $\vec{C}$ and $\vec{E}$ from knot extension.

If Z_1 and Z_2 are immediately neighbored, all three types of knot extension are equivalent. Thus we disallow $\vec{A}$ from knot extension and instead add the resulting structure $K_1M_1K_2M_2$ to the base set. Additionally this structure also needs special treatment:

Figure 6 shows all the structures that can result from extending $\vec{M}$, the white arc representing $\vec{K}$, while the gray and black arcs represent the symbols derived from $\vec{M}$. However we can also view these structures as the black arc representing $\vec{M}$, while the white and gray arcs represent symbols derived by a single extension from $\vec{K}$. Thus to arrive at an unambiguous grammar we disallow $\vec{M}$ from both stem and knot extension.

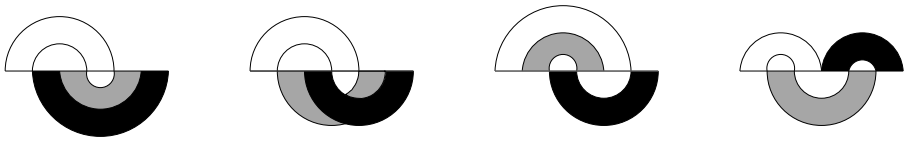

Figure 6: Extending a basic pseudoknot.

Since in case of the modified knot extension $k'_1(A)$ there are structures that can only be created by starting with a knot extension to the left we have to add the symmetric structure $M_1F_1M_2F_2$ to the base set too. There are two types of structures that can be created from both starting structures and would thus become ambiguous: Structures that can be created from $[_1]_1$ by applying both types of knot extension before the first stem extension and simple pseudoknots. The latter are created from $[_1]_1$ by only applying one type of knot extension and stem extensions of the second kind. Thus we disallow $\vec{F}$ (and the symbols derived from it) from knot extension to the right until at least one stem extension has occurred (this covers the first case of ambiguity) and assure that at least one knot extension to the right or a stem extension of the first kind is applied to $\vec{F}$ before terminating.

For $k'_2(k(A))$ we start by introducing $k(A)$ as $G_1H_1G_2H_2$ and disallowing $\vec{G}$ and $\vec{H}$ from stem extensions of the second kind since these operations are equivalent to knot extending the other symbol.

All structures for which further ambiguities arise can be created by knot extending only one of the initial symbols (the ambiguity lying in the choice which symbol is extended). Since the structures that can be generated by knot extending only one of the symbols are exactly $k'_1(A)$ we can force unambiguity by adding the rules introduced for $k'_1(A)$ and requiring both $\vec{G}$ and $\vec{H}$ to undergo at least one knot extension to the outside or preceded by a stem extension of the first kind before terminating.

Now we can apply the above rules to create unambiguous grammars for our classes:

PKF: Combining the rules for the base set A and nesting as first closure we get:

$$S \to \epsilon + US + A_1SA_2S, \quad U \to \bullet, \quad \binom{A_1}{A_2} \to \binom{[}{]},$$

which, eliminating symbols with only one associated rule (U and $\vec{A}$), can be simplified to

$$S \to \epsilon + \bullet S + [S]S.$$

L&P: Here we use the base set $k(A)$, the rules for stem extension and n' as a final closure with T the start symbol for a grammar generating A (with hooks for nesting) arriving at:

$$S \to \epsilon + U + A_1A_2 + TK_1^{(1)}K_1^{(2)}K_2^{(1)}K_2^{(2)}, \quad U \to \bullet T, \quad T \to \epsilon + \bullet T + [T]T,$$

$$\binom{A_1}{A_2} \to \binom{[T}{]T}, \quad \binom{K_1}{K_2} \to \binom{O_1I_1}{I_2O_2} + \binom{[T}{]T}, \quad \binom{O_1}{O_2} \to \binom{O_1I_1}{I_2O_2} + \binom{[T}{]T}, \quad \binom{I_1}{I_2} \to \binom{[T}{]T}.$$

Again we can eliminate symbols with only one associated rule and additionally we can replace $\vec{I}$ by $\vec{K}$ since the set of rules allowed for both symbols is the same. This results in:

$$S \to \epsilon + \bullet T + [T]T + TK_1^{(1)}K_1^{(2)}K_2^{(1)}K_2^{(2)}, \quad T \to \epsilon + \bullet T + [T]T, \quad \binom{K_1}{K_2} \to \binom{K_1[T}{]TK_2} + \binom{[T}{]T}.$$

L&P$^+$: This class diverts from L&P only by the extended base set, giving:

$$S \to \epsilon + \bullet T + [T]T + TK_1^{(1)}K_1^{(2)}K_2^{(1)}K_2^{(2)} + TK_1^{(1)}K_1^{(2)}K_1^{(3)}K_2^{(2)}K_2^{(3)}K_2^{(1)}$$
$$+TK_1^{(1)}K_1^{(2)}K_2^{(1)}K_1^{(3)}K_2^{(2)}K_2^{(3)} + TK_1^{(1)}K_1^{(2)}K_1^{(3)}K_2^{(2)}K_1^{(4)}K_2^{(3)}K_2^{(4)}K_2^{(1)},$$
$$T \to \epsilon + \bullet T + [T]T, \quad \binom{K_1}{K_2} \to \binom{K_1[T}{]TK_2} + \binom{[T}{]T}.$$

R&G: Again we use the base set $k(A)$, this time inserting S into the starting rules:

$$S \to \epsilon + US + A_1SA_2S + K_1^{(1)}SK_1^{(2)}SK_2^{(1)}SK_2^{(2)}S, \quad U \to \bullet,$$
$$\binom{K_1}{K_2} \to \binom{O_1I_1}{I_2O_2} + \binom{[}{]}, \quad \binom{O_1}{O_2} \to \binom{O_1I_1}{I_2O_2} + \binom{[}{]}, \quad \binom{I_1}{I_2} \to \binom{[}{]}.$$

Using the same simplifications as for the L&P-grammar we get:

$$S \to \epsilon + \bullet S + [S]S + K_1^{(1)}SK_1^{(2)}SK_2^{(1)}SK_2^{(2)}S, \quad \binom{K_1}{K_2} \to \binom{K_1[}{]K_2} + \binom{[}{]}.$$

C&C: Here the initial nesting is n'', thus we leave out the S after $K_1^{(2)}$. Additionally the nesting of unpaired bases is added as an outer closure:

$$S \to \epsilon + US + A_1SA_2S + K_1^{(1)}SK_1^{(2)}TK_2^{(1)}SK_2^{(2)}S, \quad T \to \epsilon + UT, \quad U \to \bullet, \quad \binom{A_1}{A_2} \to \binom{[}{]},$$
$$\binom{K_1}{K_2} \to \binom{O_1J_1}{J_2O_2} + \binom{[}{]}, \quad \binom{O_1}{O_2} \to \binom{O_1I_1}{I_2O_2} + \binom{[T}{]}, \quad \binom{I_1}{I_2} \to \binom{[T}{]T}, \quad \binom{J_1}{J_2} \to \binom{[}{]T}.$$

Again we may simplify:

$$S \to \epsilon + \bullet S + [S]S + K_1^{(1)}SK_1^{(2)}TK_2^{(1)}SK_2^{(2)}S, \quad T \to \epsilon + \bullet T,$$
$$\binom{K_1}{K_2} \to \binom{O_1[}{]TO_2} + \binom{[}{]}, \quad \binom{O_1}{O_2} \to \binom{O_1[T}{]TO_2} + \binom{[T}{]}.$$

C&C$^+$: The only difference to C&C is that now nesting of structures with paired bases after $K_1^{(2)}$ is allowed. Thus we get:

$$S \to \epsilon + \bullet S + [S]S + K_1^{(1)}SK_1^{(2)}SK_2^{(1)}SK_2^{(2)}S, \quad T \to \epsilon + \bullet T,$$
$$\binom{K_1}{K_2} \to \binom{O_1[}{]TO_2} + \binom{[}{]}, \quad \binom{O_1}{O_2} \to \binom{O_1[T}{]TO_2} + \binom{[T}{]}.$$

D&P: Compared to R&G nesting is moved from initial closure to final closure yielding:

$$S \to \epsilon + \bullet S + [S]S + K_1^{(1)}K_1^{(2)}K_2^{(1)}K_2^{(2)}, \quad \binom{K_1}{K_2} \to \binom{K_1[S}{]SK_2} + \binom{[S}{]S}.$$

A&U: Here the base class is changed. This gives (after simplifying):

$$S \to \epsilon + \bullet S + A_1 A_2, \quad \binom{A_1}{A_2} \to \binom{M_1}{K_1 M_2 K_2} + \binom{[S}{]S},$$

$$\binom{M_1}{M_2} \to \binom{M_1 K_1^{(1)}}{K_2^{(1)} K_1^{(2)} M_2 K_2^{(2)}} + \binom{K_1}{K_2}, \quad \binom{K_1}{K_2} \to \binom{K_1[S}{]SK_2} + \binom{[S}{]S}.$$

R&N: Again only the base class is changed:

$$S \to \epsilon + \bullet S + [S]S + K_1^{(1)} K_1^{(2)} K_2^{(1)} K_2^{(2)} + K_1^{(1)} K_1^{(2)} K_2^{(1)} K_1^{(3)} K_2^{(2)} K_2^{(3)} + K_1^{(1)} K_1^{(2)} K_1^{(3)} K_2^{(1)} K_2^{(2)} K_2^{(3)}$$
$$+ K_1^{(1)} K_1^{(2)} K_1^{(3)} K_2^{(1)} K_1^{(4)} K_2^{(2)} K_2^{(3)} K_2^{(4)}, \quad \binom{K_1}{K_2} \to \binom{K_1[S}{]SK_2} + \binom{[S}{]S}.$$

UMK: For this class we also have to add the rules for $k_1'(A)$:

$$S \to \epsilon + U + A_1 A_2 + K_1 M_1 K_2 M_2 + M_1 F_1 M_2 F_2, \quad U \to \bullet S, \quad \binom{A_1}{A_2} \to \binom{[S}{]S},$$

$$\binom{K_1}{K_2} \to \binom{O_1 J_1}{J_2 O_2} + \binom{Q_1 I_1}{I_2 Q_2} + \binom{C_1 B_1 C_2}{B_2} + \binom{D_1}{E_1 D_2 E_2} + \binom{[S}{]S},$$
$$\binom{O_1}{O_2} \to \binom{O_1 I_1}{I_2 O_2} + \binom{[S}{]S},$$
$$\binom{B_1}{B_2} \to \binom{O_1 J_1}{J_2 O_2} + \binom{Q_1 I_1}{I_2 Q_2} + \binom{C_1 B_1 C_2}{B_2} + \binom{D_1}{E_1 D_2 E_2} + \binom{[S}{]S},$$
$$\binom{D_1}{D_2} \to \binom{O_1 J_1}{J_2 O_2} + \binom{Q_1 I_1}{I_2 Q_2} + \binom{D_1}{E_1 D_2 E_2},$$
$$\binom{J_1}{J_2} \to \binom{C_1 B_1 C_2}{B_2} + \binom{D_1}{E_1 D_2 E_2},$$
$$\binom{Q_1}{Q_2} \to \binom{O_1 J_1}{J_2 O_2} + \binom{Q_1 I_1}{I_2 Q_2} + \binom{C_1 B_1 C_2}{B_2} + \binom{D_1}{E_1 D_2 E_2} + \binom{[S}{]S},$$
$$\binom{F_1}{F_2} \to \binom{O_1 J_1}{J_2 O_2} + \binom{Z_1 I_1}{I_2 Z_2} + \binom{C_1 F_1 C_2}{F_2},$$
$$\binom{Z_1}{Z_2} \to \binom{O_1 J_1}{J_2 O_2} + \binom{Z_1 I_1}{I_2 Z_2} + \binom{C_1 Z_1 C_2}{Z_2} + \binom{D_1}{E_1 D_2 E_2},$$
$$\binom{M_1}{M_2} \to \binom{[S}{]S}, \quad \binom{I_1}{I_2} \to \binom{[S}{]S}, \quad \binom{C_1}{C_2} \to \binom{[S}{]S}, \quad \binom{E_1}{E_2} \to \binom{[S}{]S},$$

Again we can simplify, eliminating U, $\vec{A}$, $\vec{M}$, $\vec{I}$, $\vec{C}$ and $\vec{E}$ and setting $\vec{K} = \vec{B} = \vec{Q}$:

$$S \to \epsilon + \bullet S + [S]S + K_1[SK_2]S + [SF_1]SF_2,$$

$$\binom{K_1}{K_2} \to \binom{O_1 J_1}{J_2 O_2} + \binom{K_1[S}{]SK_2} + \binom{[SK_1]S}{K_2} + \binom{D_1}{[SD_2]S} + \binom{[S}{]S},$$
$$\binom{O_1}{O_2} \to \binom{O_1[S}{]SO_2} + \binom{[S}{]S},$$
$$\binom{D_1}{D_2} \to \binom{O_1 J_1}{J_2 O_2} + \binom{K_1[S}{]SK_2} + \binom{D_1}{[SD_2]S} + \binom{[S}{]S},$$

$$\begin{pmatrix} J_1 \\ J_2 \end{pmatrix} \to \qquad\qquad \begin{pmatrix} [SK_1]S \\ K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ [SD_2]S \end{pmatrix} \qquad ,$$
$$\begin{pmatrix} F_1 \\ F_2 \end{pmatrix} \to \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \begin{pmatrix} Z_1[S \\ ]SZ_2 \end{pmatrix} + \begin{pmatrix} [SF_1]S \\ F_2 \end{pmatrix} \qquad ,$$
$$\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} \to \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \begin{pmatrix} Z_1[S \\ ]SZ_2 \end{pmatrix} + \begin{pmatrix} [SZ_1]S \\ Z_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ [SD_2]S \end{pmatrix} \qquad .$$

CCJ: Introducing the rules for $k_2'(k(A))$ and simplifying yields:

$$S \to \epsilon + \bullet S + [S]S + K_1[SK_2]S + [SF_1]SF_2 + G_1H_1G_2H_2,$$
$$\begin{pmatrix} K_1 \\ K_2 \end{pmatrix} \to \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \begin{pmatrix} K_1[S \\ ]SK_2 \end{pmatrix} + \begin{pmatrix} [SK_1]S \\ K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ [SD_2]S \end{pmatrix} + \begin{pmatrix} [S \\ ]S \end{pmatrix},$$
$$\begin{pmatrix} O_1 \\ O_2 \end{pmatrix} \to \begin{pmatrix} O_1[S \\ ]SO_2 \end{pmatrix} + \qquad\qquad \begin{pmatrix} [S \\ ]S \end{pmatrix},$$
$$\begin{pmatrix} D_1 \\ D_2 \end{pmatrix} \to \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \begin{pmatrix} K_1[S \\ ]SK_2 \end{pmatrix} + \qquad \begin{pmatrix} D_1 \\ [SD_2]S \end{pmatrix} + \begin{pmatrix} [S \\ ]S \end{pmatrix},$$
$$\begin{pmatrix} J_1 \\ J_2 \end{pmatrix} \to \qquad\qquad \begin{pmatrix} [SK_1]S \\ K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ [SD_2]S \end{pmatrix} \qquad ,$$
$$\begin{pmatrix} F_1 \\ F_2 \end{pmatrix} \to \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \begin{pmatrix} Z_1[S \\ ]SZ_2 \end{pmatrix} + \begin{pmatrix} [SF_1]S \\ F_2 \end{pmatrix} \qquad ,$$
$$\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} \to \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \begin{pmatrix} Z_1[S \\ ]SZ_2 \end{pmatrix} + \begin{pmatrix} [SZ_1]S \\ Z_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ [SD_2]S \end{pmatrix} \qquad .$$
$$\begin{pmatrix} G_1 \\ G_2 \end{pmatrix} \to \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \qquad \begin{pmatrix} [SK_1]S \\ K_2 \end{pmatrix} + \begin{pmatrix} Y_1 \\ [SY_2]S \end{pmatrix} \qquad ,$$
$$\begin{pmatrix} H_1 \\ H_2 \end{pmatrix} \to \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \qquad \begin{pmatrix} [SZ_1]S \\ Z_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ [SD_2]S \end{pmatrix} \qquad ,$$
$$\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} \to \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \qquad\qquad \begin{pmatrix} Y_1 \\ [SY_2]S \end{pmatrix} \qquad ,$$

R&E: For this class we have to replace k' by k. This gives after simplifying:

$$S \to \epsilon + \bullet S + [S]S + K_1[SK_2]S,$$
$$\begin{pmatrix} K_1 \\ K_2 \end{pmatrix} \to \begin{pmatrix} K_1I_1 \\ I_2K_2 \end{pmatrix} + \begin{pmatrix} I_1K_1I_2 \\ K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ I_1D_2I_2 \end{pmatrix} + \begin{pmatrix} L_1K_1 \\ L_2K_2 \end{pmatrix} + \begin{pmatrix} [S \\ ]S \end{pmatrix},$$
$$\begin{pmatrix} I_1 \\ I_2 \end{pmatrix} \to \qquad \begin{pmatrix} I_1K_1I_2 \\ K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ I_1D_2I_2 \end{pmatrix} + \begin{pmatrix} L_1K_1 \\ L_2K_2 \end{pmatrix} + \begin{pmatrix} [S \\ ]S \end{pmatrix},$$
$$\begin{pmatrix} D_1 \\ D_2 \end{pmatrix} \to \begin{pmatrix} K_1I_1 \\ I_2K_2 \end{pmatrix} + \qquad \begin{pmatrix} D_1 \\ I_1D_2I_2 \end{pmatrix} + \begin{pmatrix} L_1K_1 \\ L_2K_2 \end{pmatrix} + \begin{pmatrix} [S \\ ]S \end{pmatrix},$$
$$\begin{pmatrix} L_1 \\ L_2 \end{pmatrix} \to \begin{pmatrix} K_1I_1 \\ I_2K_2 \end{pmatrix} + \begin{pmatrix} I_1K_1I_2 \\ K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ I_1D_2I_2 \end{pmatrix} + \qquad \begin{pmatrix} [S \\ ]S \end{pmatrix}$$

4.2 Enumeration

4.2.1 A Technical Lemma

Let $G = (I, d, T, P, S)$ a MCFG and $\mathcal{T}$ an ordered tree. We call $\mathcal{T}$ *a derivation tree for* $w \in T^\star$ *and* G iff

1. the root of $\mathcal{T}$ is labeled S;
2. the frontier of $\mathcal{T}$ is given by w;
3. any internal node η is labeled by a component X_i of an intermediate symbol $\vec{X}$; for the successors of η we demand the existence of a production $f : \vec{X} \to \vec{\alpha}$ with
   - the direct successors of η from left to right are labeled by the symbols of α_i, X_i the label of η;
   - for all internal nodes η' that correspond to different components of the same intermediate (not only by name but with respect to the step in which it has been *generated*) the same production f is used in connection with the before item.

If we start the derivation with an intermediate $\vec{X}$ of dimension $d(\vec{X})$ where the components are ordered (from left to right) $X_{i_1}, X_{i_2}, \ldots, X_{i_{d(\vec{X})}}$, then we have to consider an ordered forest of $d(\vec{X})$ derivation trees where the root of the j-th tree $\mathcal{T}_j$ is labeled X_{i_j}, $1 \leq j \leq d(\vec{X})$.
Like in the regime of context-free grammars, we define for a MCFG G the degree of ambiguity $deg(w)$ of $w \in \mathcal{L}(G)$ by the number of different derivation trees for w. By $deg_{X_{i_1}, X_{i_2}, \ldots, X_{i_{d(X)}}}(w)$ we denote the number of different derivation forests for w starting with intermediate $\vec{X}$ – we will say that the forest is rooted by $\vec{X}$. We call G ambiguous if there exists $w \in \mathcal{L}(G)$ with $deg(w) > 1$. We write $deg^G_\circ(w)$ for the degree of ambiguity of word w with respect to grammar G using index $\circ$ as introduced before. To algebraically determine the degree of ambiguity we start with the following observation:

Lemma 4.1. *Let* $G = (I, d, T, P, S)$ *a* MCFG *and* $\vec{X} \in I$. *Then*

$$\sum_{w \in T^n} deg_{X_{i_1}, X_{i_2}, \ldots, X_{i_{d(\vec{X})}}}(w) = \sum_{w' \in T^n} deg_{X_1, X_2, \ldots, X_{d(\vec{X})}}(w').$$

Proof: For any derivation forest $\mathcal{T}_i$, $1 \leq i \leq d(\vec{X})$, with frontier $w \in T^n$ and the sequence of root labels $X_{i_1}, X_{i_2}, \ldots, X_{i_{d(\vec{X})}}$ we have $w_i \in T^\star$ such that

- $w_1 \cdot w_2 \cdots w_{d(\vec{X})} = w$, and
- tree $\mathcal{T}_i$ has frontier w_i.

To switch from $deg_{X_{i_1}, X_{i_2}, \ldots, X_{i_{d(\vec{X})}}}$ to $deg_{X_1, X_2, \ldots, X_{d(\vec{X})}}$ we only have to permute the ordering of the $\mathcal{T}_i$ such that the frontier of the resulting reordered derivation forest is given by a permutation of the w_i, $1 \leq i \leq d(\vec{X})$, of the same total length n. Thus the original derivation forest of w rooted $X_{i_1}, X_{i_2}, \ldots, X_{i_{d(\vec{X})}}$ implies a derivation forest for $w' \in T^n$ rooted $X_1, X_2, \ldots, X_{d(\vec{X})}$. Obviously, this reasoning applies to both directions proving equality. □

Theorem 4.2. *Let* $G_m = (I_m, d, T, P_m, S)$ *an unambiguous* MCFG. *There is a* CFG $G = (I, T, P, S)$ *with*

$$|\mathcal{L}(G_m) \cap T^n| = \sum_{w \in \mathcal{L}(G) \cap T^n} deg_G(w)$$

for all $n \in \mathbb{N}_0$.

Proof: We assume $I_m = \{\vec{X}^{<1>}, \ldots, \vec{X}^{<k>}\}$. For $n \in \mathbb{N}_0$ let $\mathbf{X}^{<k>}(n)$ denote the number of different derivation forests for words of length n rooted by $\vec{X}^{<k>}$. By the previous lemma we conclude that this number is independent of the ordering assumed for the components of $\vec{X}^{<j>}$ (assumed for the trees of the corresponding forest). Since by assumption G_m is unambiguous, $\mathbf{X}^{<1>}(n)$ equals $|\mathcal{L}(G_m) \cap T^n|$. We use R_j to denote the set of productions with left-hand side $\vec{X}^{<j>}$, for $f \in P_m$ we write $t(f)$ to represent the number of terminal symbols to be found within the right-hand side of f and $i(f)$ to denote the multi-set of intermediates showing up on the right-hand side of f. Then the number of different derivation forests for a word of length n rooted by $\vec{X}^{<j>}$ is obviously given by the recurrence relation

$$\mathbf{X}^{<j>}(n) = \sum_{f \in R_j} \sum_{n_1+n_2+\cdots+n_{|i(f)|}=n-t(f)} \prod_{1 \le j \le |i(f)|} \mathbf{X}^{<i_j>}(n_j)$$

since we have to choose one production f from R_j for the first step of the derivation which contributes $t(f)$ terminals such that the remaining $n - t(f)$ symbols of a word of length n have to be derived from the newly produced intermediates $X^{<i_j>} \in i(f)$ allowing any partition of $n - t(f)$ into contributions n_j for the various[4] $X^{<i_j>}$. Here we assume $n_i \in \mathbb{N}_0$, $1 \le i \le |i(f)|$, and $\sum_{n_1+n_2+\cdots+n_{|i(f)|}=n-t(f)} \cdots = \delta_{n,t(f)}$, δ Kronecker's symbol, for $i(f) = \emptyset$. Accordingly, the ground cases of this recursion are given by productions f without intermediate symbols on their right-hand sides which contribute 1 in cases where $t(f) = n$ holds. If we multiply this equation by z^n and sum over all n we find the following equation for the ordinary generating functions $\mathcal{X}^{<j>}(z) := \sum_{n \ge 0} \mathbf{X}^{<j>}(n) z^n$

$$\mathcal{X}^{<j>}(z) = \sum_{f \in R_j} z^{t(f)} \cdot \prod_{1 \le l \le |i(f)|} \mathcal{X}^{<i_l>}(z)$$

by making use of series convolution. According to [14] after obvious simplifications (equations like $X = A$, $A = \alpha$ with no further occurrence of A are simplified to $X = \alpha$) the same system of equations counting the number of different derivation trees, i.e., the degree of ambiguity of all words in $\mathcal{L}(G)$, results from the CFG G with the set of productions P given by[5]

$$\bigcup_{1 \le j \le k} \bigcup_{f \in R_j} \left\{ \mathsf{X}^{<j>} \to \mathsf{A}_f, \mathsf{A}_f \to a^{t(f)} \bigodot_{1 \le l \le |i(f)|} \mathsf{X}^{<i_l>} \right\}.$$

Here a is an arbitrary terminal symbol and $I := \bigcup_{f \in P_m} \{\mathsf{A}_f\} \cup \bigcup_{1 \le j \le k} \mathsf{X}^{<j>}$. This context-free grammar may not be unambiguous. However since the identical system of equations for its generating functions is associated with the degree of ambiguity of all words of the same length and not with the number of words generated, the theorem follows. □

Remark: Our construction from the previous proof implicitly provides a bijection between the objects described by a MCFG and a plain context-free encoding. This is the idea of bijective combinatorics used, e.g., in [24], to enumerate some RNA pseudoknot classes. However, this classic approach – even if providing additional insight by the explicit knowledge of a bijection – cannot be used in cases where no such bijection is known (like for the R&E or the R&N class). We therefore assume our findings a mayor progress for cases where such a bijection is out of reach.

Corollary 4.3. *Let $G_m = (I, d, T, P, \vec{X}^{<1>})$, $I = \{\vec{X}^{<1>}, \ldots, \vec{X}^{<k>}\}$, an unambiguous* MCFG *without ε-rules[6] and $\mathcal{SE}$ the system of equations where for each $\vec{X}^{<i>} \in I$ the following variable*

[4] Note that if the same symbol shows up l times on the right-hand side of f then the corresponding factor is l times part of the product; it is for this reason we defined $i(f)$ as a multi-set.

[5] We introduce the new intermediates A_f to be able to distinguish cases where different productions f, f' from R_j introduce the same string $a^{t(f)} \bigodot_{1 \le l \le |i(f)|} \mathsf{X}^{<i_l>}$.

[6] For MCFGs an ε-rule is given by a rule like $\vec{A} \to (\varepsilon, \ldots, \varepsilon)^T$.

and corresponding equation is introduced:

$$X^{(i)}(z) = \sum_{\vec{\alpha}:\vec{X}^{<i>}\to\vec{\alpha}\in P} \prod_{1\le j\le d(X^{<i>})} h(\alpha_j), \tag{1}$$

where h is the substitution that maps $a \in T$ to variable z, the first component $X_1^{<j>}$ of any intermediate symbol to $X^{(j)}(z)$ and its other components to 1 (replacing concatenation of symbols by multiplication). Solving $\mathcal{SE}$ for $X^{(1)}(z)$, choosing the unique solution compatible with initial conditions, $[z^n]X^{(1)}(z) = |\mathcal{L}(G) \cap T^n|$ holds.

Proof: This corollary is immediate from the previous theorem and its proof showing that the system of equations (1) results from enumerating the number of words in $\mathcal{L}(G_m) \cap T^n$ and at the same time from enumerating the degree of ambiguity of the words generated by a context-free grammar G. For the latter it is well known (see ,e.g., [14]) that it has a unique solution compatible with initial conditions fulfilling $[z^n]X^{(1)}(z) = |\mathcal{L}(G_m)\cap T^n| = \sum_{w\in\mathcal{L}(G)\cap T^n} deg_G(w)$ in cases where G has no ε-rules (which is guaranteed by assuming G_m to have no ε-rules). □

Remarks: The assumption of ε-freeness is used in [14] to prove convergence of a sequence of formal power series to the solution of the system of equations. However, this sequence may also be convergent for non-ε-free grammars in which case the corollary still applies (this, e.g., holds for our application presented in Section 4.2.2). Otherwise, we may use standard techniques to transform a grammar into an equivalent ε-free one.

4.2.2 Application

We will use the grammars of Section 4.1 to enumerate the different pseudoknot classes considering two different notions of size: first every base (terminal symbol) is counted and second – in the style of [24] – the size is given by the number of base pairs ignoring unpaired bases. Since the computations needed for enumeration are always the same we will demonstrate our approach using the R&E class only. Nevertheless, results will be provided for all classes (see Table 3).
Starting with the simplified grammar for class R&E and applying Corollary 4.3 yields the following system of equations

$$\begin{aligned}
S(z) &= 1 + z\cdot S(z) + z^2\cdot S(z)^2 + z^2\cdot S(z)^2\cdot K(z),\\
K(z) &= I(z)\cdot K(z) + I(z)\cdot K(z) + D(z)\cdot I(z) + L(z)\cdot K(z) + z^2\cdot S(z)^2,\\
I(z) &= I(z)\cdot K(z) + D(z)\cdot I(z) + L(z)\cdot K(z) + z^2\cdot S(z)^2,\\
D(z) &= I(z)\cdot K(z) + D(z)\cdot I(z) + L(z)\cdot K(z) + z^2\cdot S(z)^2,\\
L(z) &= I(z)\cdot K(z) + I(z)\cdot K(z) + D(z)\cdot I(z) + z^2\cdot S(z)^2.
\end{aligned}$$

This system of equations can be reduced to a single equation for $S(z)$ yielding

$$\begin{aligned}
0 = {} & S(z)(-6+12S(z)-6S(z)^2)z + S(z)^2(-10+14S(z)-4S(z)^2)z^2\\
& + S(z)^3(-10+8S(z))z^3 + S(z)^4(-4-S(z)+S(z)^2)z^4 + (1-2S(z))S(z)^5z^5\\
& + 2S(z)^6z^6 - (2-6S(z)+6S(z)^2-2S(z)^3).
\end{aligned}$$

At this point we proceed along the lines of [12] in order to determine an expansion of $S(z)$ at its dominant singularity from the implicit representation given before. For this purpose we first compute the set of critical points for $S(z)$. Among those we select the one of smallest modulus which is connected to a singularity getting $d = 0.2010468602\ldots$ By substituting $S = 1.3605149159\cdots + Y$ and $z = 0.2010468602\cdots - Z$ the singularity (a branching point) is shifted to the origin of the (Y, Z)-plane where we now are able to read off the exponent $\frac{1}{2}$ of Z for the expansion of Y at d. Using the ansatz $c\cdot\sqrt{Z}$ for the expansion of Y it becomes possible to determine c; re-substituting while applying the $\mathcal{O}$-transfer method [9] finally yields

$$[z^n]S(z) \sim \frac{0.0348606347\ldots d^{-n}}{2\sqrt{\pi n^3}} + o(d^{-n}n^{-3/2})$$

Class	Asymptotic	size = arcs		size = all		Run time of prediction alg.
		α	ω	α	ω	
PKF	$\frac{\alpha}{2\sqrt{\pi n^3}}\omega^n$	2	4	3.9149	2.6180	$\mathcal{O}(n^3)$
L&P	$\alpha\omega^n$	$\frac{1}{2}$	4	$\frac{1}{4}$	3	$\mathcal{O}(n^5)$
L&P$^+$	$\alpha n\omega^n$	$\frac{1}{8}$	4	$\frac{1}{48}$	3	$\mathcal{O}(n^5)$
R&G	$\frac{\alpha}{2\sqrt{\pi n^3}}\omega^n$	1.1651	6.576	2.7058	3.5129	$\mathcal{O}(n^4)$
C&C	$\frac{\alpha}{2\sqrt{\pi n^3}}\omega^n$	1.6651	5.857	3.9560	3.4201	$\mathcal{O}(n^6)$
C&C$^+$	$\frac{\alpha}{2\sqrt{\pi n^3}}\omega^n$	1.1651	6.576	2.7003	3.5644	$\mathcal{O}(n^6)$
D&P	$\frac{\alpha}{2\sqrt{\pi n^3}}\omega^n$	0.7535	7.315	1.7082	3.7046	$\mathcal{O}(n^5)$
A&U	$\frac{\alpha}{2\sqrt{\pi n^3}}\omega^n$	0.6575	7.547	1.4813	3.7472	$\mathcal{O}(n^5)$
R&N	$\frac{\alpha}{2\sqrt{\pi n^3}}\omega^n$	0.6429	8.284	1.4222	3.8782	$\mathcal{O}(n^6)$
UMK	$\frac{\alpha}{2\sqrt{\pi n^3}}\omega^n$	0.4719	8.912	1.0295	3.9854	$\mathcal{O}(n^5)$
CCJ	$\frac{\alpha}{2\sqrt{\pi n^3}}\omega^n$	0.4432	9.151	0.9619	4.0250	$\mathcal{O}(n^5)$
R&E	$\frac{\alpha}{2\sqrt{\pi n^3}}\omega^n$	0.0176	15.792	0.0348	4.9739	$\mathcal{O}(n^6)$

Table 3: The asymptotical number of pseudoknot structures of size n.

where $d^{-1} = 4.9739647694\ldots$ holds.
If we now want the size of a structure to be given by its number of base pairs, we slightly have to tweak the application of the Corollary: When translating the grammar to a system of equations not every terminal symbol is replaced by z but only every opening bracket. All the remaining parts of the analysis are left unchanged and are therefore not explained more detailed.

5 Conclusion

In this paper we provided a unifying algebraic view on the *zoo* of pseudoknot classes which originally have been introduced by various, typically application-driven concepts. That way, all the different classes get comparable for the first time. Making use of operations that act on the dot bracket representation of the pseudoknots in order to generate different structural motives our approach is rather generic such that classes to be invented in the future can easily be integrated. A first application of our characterization comes with recognition algorithms presented in this paper, which allow to decide in linear time whether or not a pseudonot (dot bracket word) belongs to a given class. We offer a corresponding web-service to the scientific community (see `http://wwwagak.cs.uni-kl.de/KnotFilter` for details). Starting at our algebraic description it becomes possible to derive unambiguous multiple context-free grammars that generate the different classes. We used these grammars to determine the precise asymptotic number of pseudoknots building on a new lemma interrelating multiple context-free grammars and generating functions.

That way, the problem of enumerating the Rivas & Eddy class – open since more than 10 years – was solved; many further original enumerating results have also been proven. It is worth mentioning that this technique is not restricted to applications in the domain of RNA pseudoknots but allows usage for any kind of discrete objects that can be modeled by unambiguous multiple context-free languages.

Furthermore, our grammars may be used to derive stochastic models of RNA pseudoknots or for the prediction of RNA structure based on them ([13] contains similar ideas).

Finally, they may be useful to determine partition functions for pseudoknot structures for our grammars imply a recursive decomposition of the structures along the different structural motifs.

References

[1] T. Akutsu. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discr. Appl. Math.*, 104:45–62, 2000.

[2] J.J. Cannone, S. Subramanian, M.N. Schnare, J.R. Collett, L.M. D'Souza, Y. Du, B. Feng, N. Lin, L.V. Madabusi, K.M. M?ller, N. Pande, Z. Shang, N. Yu, and R.R. Gutell. The comparative RNA web (CRW) site: An online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BioMed Central Bioinformatics*, 3(2), 2002.

[3] Song Cao and Shi-Jie Chen. Predicting structures and stabilities for h-type pseudoknots with interhelix loops. *RNA*, 15(4):696–706, 2009.

[4] Ho-Lin Chen, Anne Condon, and Hosna Jabbari. An $O(n^5)$ algorithm for MFE prediction of kissing hairpins and 4-chains in nucleic acids. *J. Comp. Biol.*, 16:803–815, 2009.

[5] S. J. Chen. RNA folding: conformational statistics, folding kinetics, and ion electrostatics. *Annu Rev Biophys*, 37:197–214, 2008.

[6] Anne Condon, Beth Davy, Baharak Rastegari, Shelly Zhao, and Finbarr Tarrant. Classifying RNA pseudoknotted structures. *Theor. Comp. Sci.*, 320:35–50, 2004.

[7] R. M. Dirks and N. A. Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *J. Comput. Chem.*, 24:1664–1677, 2003.

[8] J A Doudna and T R Cech. The chemical repertoire of natural ribozymes. *Nature*, 418:222–228, 2002.

[9] Philippe Flajolet and Andrew Odlyzko. Singularity analysis of generating functions. *SIAM J. Disc. Math.*, 3:216–240, 1990.

[10] D P Giedroc and P V Cornish. Frameshifting RNA pseudoknots: structure and mechanism. *Virus Res.*, 139:193–208, 2009.

[11] Michael A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley Publishing Company, 1978.

[12] Einar Hille. *Analytic Function Theory, Volume II.* Chelsea Publishing Company, 1962.

[13] Yuki Kato, Hiroyuki Seki, and Tadao Kasami. RNA pseudoknotted structure prediction using stochastic multiple context-free grammar. *IPSJ Digital Courier*, 2:655–664, 2006.

[14] Werner Kuich and Arto Salomaa. *Semirings, automata, languages.* Springer London, 1986.

[15] Rune B. Lyngsø and Christian N. Pedersen. Pseudoknots in RNA secondary structures. In *Proceedings of the 4th Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 201–209. ACM Press, 2000.

[16] Rune B. Lyngsø and Christian N. Pedersen. RNA pseudoknot prediction in energy-based models. *J. Comp. Biol.*, 7:409–427, 2000.

[17] Hiroshi Matsui, Kengo Sato, and Yasubumi Sakakibara. Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures. *Bioinformatics*, 21:2611–2617, 2005.

[18] Dirk Metzler and Markus E. Nebel. Predicting RNA secondary structures with pseudoknots by MCMC sampling. *J. Math. Biol.*, 56:161–181, 2008.

[19] Olivier Namy, Stephen J. Moran, David I. Stuart, Robert J. C. Gilbert, and Ian Brierley. A mechanical explanation of RNA pseudoknot function in programmed ribosomal frameshifting. *Nature*, 441:244–247, 2006.

[20] Jens Reeder and Robert Giegerich. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics*, 5:104, 2004.

[21] Christian M. Reidys, Fenix W.D. Huang, Jørgen E. Andersen, Robert C. Penner, Peter F. Stadler, and Markus E. Nebel. Topology and prediction of RNA pseudoknots. *Bioinformatics*, 2011.

[22] Elena Rivas and Sean R. Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol.*, 285:2053–2068, 1999.

[23] Einar Andreas Rødland. Pseudoknots in RNA secondary structures: Representation, enumeration, and prevalence. *J. Comp. Biol.*, 13:1197–1213, 2006.

[24] Cédric Saule, Mireille Régnier, Jean-Marc Steyaert, and Alain Denise. Counting RNA pseudoknotted structures. *Journal of Computational Biology*, to appear.

[25] Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao. Kasami. On multiple context free grammars. *Theor. Comp. Sci.*, 88:191–229, 1991.

[26] David W Staple and Samuel E Butcher. Pseudoknots: RNA structures with diverse functions. *PLoS Biol.*, 3:e213, 2005.

[27] Michela Taufer, Abel Licon, Roberto Araiza, David Mireles, F. H. D. van Batenburg, Alexander Gultyaev, and Ming-Ying Leung. `PseudoBase++`: an extension of `PseudoBase` for easy searching, formatting and visualization of pseudoknots. *Nucleic Acids Res.*, 37:D127–D135, 2009.

[28] C A Theimer, C A Blois, and J Feigon. Structure of the human telomerase RNA pseudoknot reveals conserved tertiary interactions essential for function. *Mol. Cell*, 17:671–682, 2005.

[29] A. Uemura Y., Hasegawa, S. Kobayashi, and T. Yokomori. Tree adjoining grammars for RNA structure prediction. *Theor. Comp. Sci.*, 210:277–303, 1999.

[30] Sebastian Wild. An earley-style parser for solving the rna-rna interaction problem. Bachelor thesis, TU Kaiserslautern, 2010.

[31] M. Zuker and P. Stiegler. Optimal computer folding of larger RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.*, 9:133–148, 1981.