

Algebraic and Combinatorial Properties of Common RNA Pseudoknot Classes with Applications

MARKUS E. NEBEL and FRANK WEINBERG

ABSTRACT

Predicting RNA structures with pseudoknots in general is an NP-complete problem. Accordingly, several authors have suggested subclasses that provide polynomial time prediction algorithms by allowing (respectively, disallowing) certain structural motives. In this article, 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 such as enumerating the Rivas & Eddy class of pseudoknots.

Key words: algorithms, combinatorics, RNA, secondary structure.

1. INTRODUCTION

TO A LARGE EXTENT THE FUNCTION OF AN RNA MOLECULE is determined by its secondary structure, that is, 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 (Taufer et al., 2009) lists more than 300 records of pseudoknots determined by a variety of experimental and computational techniques, including crystallography, NMR spectroscopy, 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 (Doudna and Cech, 2002), telomerase activity (Theimer et al., 2005), reviewed in Staple and Butcher (2005) and Giedroc and Cornish (2009), and programmed frameshifting (Namy et al., 2006). 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 (Akutsu, 2000; Lyngsø and Pedersen, 2000b). However, polynomial-time algorithms have been devised for certain restricted classes of pseudoknots. The following references provide a certainly incomplete list of those approaches: Akutsu (2000); Cao and Chen (2009); Chen et al. (2009); Dirks and Pierce (2003); Kato et al. (2006); Lyngsø and Pedersen (2000a); Matsui et al. (2005); Reeder and Giegerich (2004); Reidys et al. (2011); Rivas and

Eddy (1999); and Uemura Y. et al. (1999). The interrelationships of some of these classes have been partially clarified by Condon et al. (2004) and Rødland (2006). Furthermore, Saule et al. (2011) presents enumeration results quantifying the size of some of those classes. In addition to the previously mentioned exact algorithms, plenty of heuristic approaches to pseudoknot prediction have been suggested; see, e.g., Chen (2008), Metzler and Nebel (2008), 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 representations for all the pseudoknot classes are found. By proving a new enumeration lemma that allows for computing 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

2.1. 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 such 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 entirety of this article 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.

2.2. The Rivas and Eddy (R&E) class

In Rivas and Eddy (1999) an extension of the traditional dynamic programming scheme of Zuker and Stiegler (1981) is introduced, which allows for predicting a huge class of pseudoknots.

The original algorithm basically determines the optimal folding for any connected region of the RNA molecule by considering either adding a pairing consisting of the first and last base of the region to the optimal folding of the remaining region, or splitting 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.

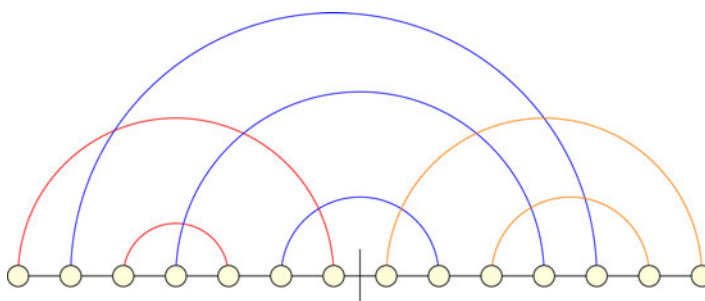
The new algorithm additionally considers regions that are not entirely connected but have a single gap inside. Figure 1 shows the different ways a connected or gapped region may be decomposed into two gapped regions. Here the horizontal lines show (segments of) the backbone of the molecule and a semi-disc in the upper (respectively, 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. 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 is a single base pair.

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 Condon et al. (2004) gives a linear time algorithm that decides if



FIG. 1. Decomposing a connected or gapped region into two gapped regions.

FIG. 2. A UMK-pseudoknot.



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.

2.3. The Uemura et al. (UMK) class

In Uemura et al. (1999), a set of rules for tree-adjoined grammars is suggested that can be used to generate a language that models pseudoknots. In Matsui et al. (2005), these rules are translated into the setting of pair-stochastic tree-adjoined grammar. In Kato et al. (2006), an equivalent set of rules for stochastic multiple context-free grammars is given. 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 either 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.

2.4. The Akutsu and Uemura (A&U) class

In Akutsu (2000), a minimum free energy-based dynamic programming algorithm is proposed that predicts pseudoknots from a subclass of UMK in $\mathcal{O}(n^5)$ time.

A&U structures are built from so-called simple pseudoknots. A simple pseudoknot consists of two groups of base pairs, the base pairs of each group forming 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 Condon et al. (2004) and extended in Saule et al. (2011), where additionally its size was determined. A linear time recognition algorithm, however, has not yet been given.

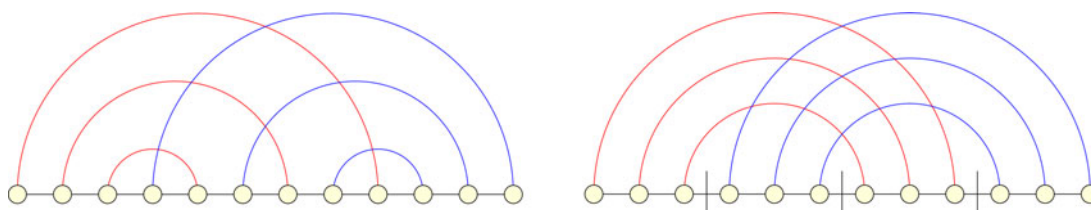


FIG. 3. A simple pseudoknot (left) and an H-type pseudoknot (right).

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

In Lyngsø and Pedersen (2000a), an algorithm based on the minimum free energy paradigm is presented that runs in $\mathcal{O}(n^5)$ time and $\mathcal{O}(n^3)$ space and predicts structures in 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 Condon et al. (2004). Additionally, Condon et al. (2004) also defines the subclass of structures $w_1u_1w_2u_2$, w_1w_2 and u_1u_2 pseudoknot free, and labels that class as L&P.

For L&P, Condon et al. (2004) gives a linear time recognition algorithm along with some inclusion results, and Saule et al. (2011) extends the inclusion results and gives the size of the class. $L\&P^+$ has not yet been further examined.

2.6. The Dirks and Pierce (D&P) class

In Dirks and Pierce (2003), a dynamic programming algorithm is introduced 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 Condon et al. (2004) and Saule et al. (2011).

2.7. The Reeder and Giegerich (R&G) class

In Reeder and Giegerich (2004), an algorithm is presented 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 Saule et al. (2011).

2.8. The Cao and Chen (C&C) classes

In Cao and Chen (2009), another algorithm for predicting H-type pseudoknots is given. This algorithm is based on a more sophisticated energy model 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 Cao and Chen (2009) if embedding substructures at the middle one of the marked positions is allowed. In Saule et al. (2011), the class in which this is forbidden is named 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 Saule et al. (2011).¹

2.9. The Chen, Condon, and Jabbari (CCJ) class

In Chen et al. (2009), 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 so-called TGB gapped structures that are combined as in the first 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 Chen et al. (2009) but neither a recognition algorithm nor the size of the class are known.

¹In Saule et al. (2011), unpaired bases are ignored, making $C\&C^+$ identical to R&G in their setting.

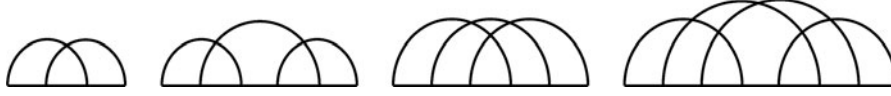


FIG. 4. All shadows with topological genus 1.

2.10. The Reidys and Nebel (R&N) class

The approach of Reidys et al. (2011) 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 Reidys et al. (2011), but neither the size of the class nor a 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

$$B = n'(C, C') := C \cup \{w_1 u w_2 \mid w = w_1 w_2 \in C, u \in C', u \perp w\} \text{ or}$$

$$B = n(C) := n'(C, C)$$

$$B = n''(C) := C \cup \{w_1 u w_2 \mid w = w_1 w_2, u \in C, u \perp w, (w_1, w_2) = (w'_1 [i,]_j w'_2) \Rightarrow i = j\}.$$

- *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

$$B = k(C) := C \cup \{w_1 [j [i]_i]_j w_2]_i w_3, w_1 [i w_2 [j]_j]_i w_3, w_1 [i [j w_2]_j]_i w_3 \mid [j]_j \perp w_1 [i w_2]_i w_3 \in C\} \text{ or}$$

$$B = k'_l(C) := C \cup \{w_1 [j [i]_i]_j w_2]_i w_3, w_1 [i w_2 [j]_j]_i w_3 \mid w = w_1 [i w_2]_i w_3 \in C, i \leq l, w \perp [j]_j\}.$$

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 $n'^\infty(C, C') := n'(n'(\dots n'(C, C'), \dots, C'), C')$.

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 $U = A \setminus \{[1]_1\}$, $L = k(A) \cup \{[1]_2[3]_2[3]_1, [1]_2[1]_3[2]_3, [1]_2[3]_2[4]_3[4]_1\}$, $Si = A \cup \{s \mid s \text{ is shadow of a simple pseudoknot}\}$ and $G = A \cup \{s \mid s \text{ is shadow of a genus-1-pseudoknot}\}$, we find:

$$\begin{aligned}
 PKF &= n^\infty(A), & D\&P &= (s \circ n)^\infty(k(A)), \\
 L\&P &= n'^\infty(s^\infty(k(A)), A), & A\&U &= (s \circ n)^\infty(Si), \\
 L\&P^+ &= n'^\infty(s^\infty(L), A), & R\&N &= (s \circ n)^\infty(G), \\
 R\&G &= s^\infty(n^\infty(k(A))), & U\&M\&K &= (s \circ n \circ k'_1)^\infty(A), \\
 C\&C &= n'^\infty(s^\infty(n''^\infty(k(A))), U), & C\&C\&J &= (s \circ n \circ k'_2)^\infty(k(A)), \\
 C\&C^+ &= n'^\infty(s^\infty(n^\infty(k(A))), U), & R\&E &= (s \circ n \circ k)^\infty(A).
 \end{aligned}$$

From these descriptions we can easily conclude the inclusion hierarchy of the classes:

Lemma 3.1. (illustrated in Fig. 5)

- $PKF \subset R\&G \subset C\&C^+ \subset D\&P \subset A\&U \subset U\&M\&K \subset C\&C\&J \subset R\&E$,
- $PKF \subset L\&P \subset L\&P^+ \subset R\&N \subset R\&E$,
- $L\&P^+ \subset U\&M\&K$,
- $PKF \subset C\&C \subset C\&C^+$,
- $L\&P \subset D\&P \subset R\&N$,
- $C\&C \subset R\&G$ 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 no crossing parentheses we find $n^\infty(A) = n''^\infty(A)$ giving $PKF \subset C\&C$.

The elements of Si can be generated from A by alternately 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 $A\&U \subset U\&M\&K$.

Listing all elements of the latter shows $G \subset k^3(A)$. Additionally $k(A) \subset L \subset n(G)$, thus $D\&P \subset R\&N \subset R\&E$ and $L\&P^+ \subset R\&N \cap U\&M\&K$. The other inclusions are immediate.

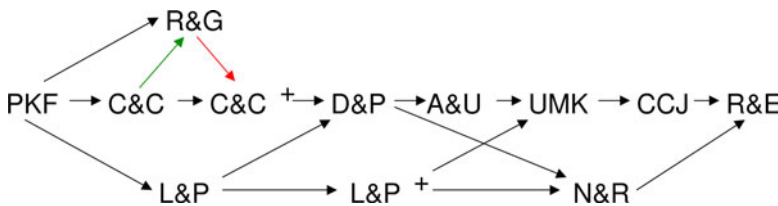


FIG. 5. Hierarchy of pseudoknot classes.

For the incomparability results note that

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

Finally, the fact that the inclusions are proper follows from the size results listed in Section 4.2.2 (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.

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  $w = w_1[_j[_i] w_2]_{i,p} w_3$ 
    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  $k'_m(B)$ ,  $w = w_1[_j[_i] w_2]_{i,p} w_3$ ,  $type[i] \neq restricted$ 
    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  $w = w_1[_i[_k w_2[_l]_k]_j]_{l,p} w_3$  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

```

TABLE 1. COVERAGE OF BIOLOGICAL STRUCTURES

<i>Class</i>	<i>PBase</i>	<i>16S</i>	<i>23S</i>	<i>Grp I Intron</i>	<i>Grp II Intron</i>	<i>Overall [%]</i>
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

Theorem 3.2. *For C a class from Section 2 and w a word, Algorithm 1 decides if $w \in C$.*

Proof. Due to space restrictions this proof had to be omitted from this version of the paper. It can be found in an extended version, together with a link to the web service, in the Supplementary Material available online at www.liebertonline.com/cmb. ■

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 (Condon et al., 2004), each iteration can be done in $\mathcal{O}(1)$, and the complete algorithm will have linear runtime in the length of w .

3.2. Coverage of biological structures

We used Algorithm 1 to test the various classes for their coverage of biological structures in PseudoBase (PBase) (Taufer et al., 2009) as well as 16S and 23S rRNA and Group I and II Introns from the Gutell database (Cannone et al., 2002). The results are listed in Table 1.

Since a number of the database entries contains isolated base pairs, i.e., base pairs that 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.

TABLE 2. COVERAGE OF BIOLOGICAL STRUCTURES WITH ISOLATED BASE PAIRS REMOVED

<i>Class</i>	<i>PBase</i>	<i>16S</i>	<i>23S</i>	<i>Grp I Intron</i>	<i>Grp II Intron</i>	<i>Overall [%]</i>
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

4. GRAMMARS

It is a simple exercise to use Ogden's Lemma (Harrison, 1978) to prove that the language describing any class of pseudoknotted structures is not context-free. As a consequence, in order to enumerate classes 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 (Seki et al., 1991; Wild, 2010). 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 nonterminal $A \in I$ and an index $i \in \{1, \dots, d(A)\}$, we write A_i for the i th component of A . Restricting $d(A) = 1$ for all nonterminals A yields a plain CFG. The general form of a multi-dimensional rule is again quite similar to CFGs, except that it is in terms of “vectors” instead of “scalars”:

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

Often it will be convenient to denote the premise (respectively, conclusion) of such a rule by \vec{A} or A (respectively $\vec{\alpha}$ 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 $A_i^{(1)}, A_i^{(2)}, \dots$ 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 terminal strings around and between 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 Seki et al., 1991). 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 $\Rightarrow_{lm} \subseteq (T \cup \mathcal{I})^* \times (T \cup \mathcal{I})^*$ for MCFGs is then defined by $xA_{i_1}\beta_1A_{i_2}\beta_2 \cdots A_{i_{d(A)}}\beta_{d(A)} \Rightarrow_{lm} x\alpha_{i_1}\beta_1\alpha_{i_2}\beta_2 \cdots \alpha_{i_{d(A)}}\beta_{d(A)}$ iff $x \in T^*$, $\vec{A} \rightarrow \vec{\alpha} \in P$, $\{i_1, i_2, \dots, i_{d(A)}\} = \{1, 2, \dots, d(A)\}$, and the components $A_{i_1}, A_{i_2}, \dots, A_{i_{d(A)}}$ were introduced in the same derivation step as belonging together. So, leftmost derivation in MCFGs means expanding the intermediate that 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_{lm}^* to be the reflexive and transitive closure of \Rightarrow_{lm} and are finally able to define the language of an MCFG G : $\mathcal{L}(G) := \{w \in T^* \mid S \Rightarrow_{lm} w\}$.

4.1. Translating algebraic specifications into MCFGs

Now we will transform the specifications from Section 3 into unambiguous MCFGs. 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 $\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix}$ and each \bullet 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 $\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} \rightarrow \begin{pmatrix} \mid \\ \mid \end{pmatrix}$ and $Y \rightarrow \bullet$ for all applicable symbols. Note that to avoid the necessity of an infinite alphabet, we abstract from the indices and encode the information that bases form pairs only in the derivation tree. This is acceptable since the applications of unambiguous grammars we are aiming at actually rely on a 1-to-1 correspondence between derivation trees and structures that 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 $S \rightarrow \epsilon$, $S \rightarrow U$ and $S \rightarrow A_1 A_2$, for $k(A)$ we add the rule $S \rightarrow K_1^{(1)} K_1^{(2)} K_2^{(1)} K_2^{(2)}$. For L and G , 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: $\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \rightarrow \begin{pmatrix} M_1 \\ K_1 M_2 K_2 \end{pmatrix}$, $\begin{pmatrix} M_1 \\ M_2 \end{pmatrix} \rightarrow \begin{pmatrix} N_1 K_1 \\ K_2 N_2 \end{pmatrix}$, $\begin{pmatrix} N_1 \\ N_2 \end{pmatrix} \rightarrow \begin{pmatrix} M_1 \\ K_1 M_2 K_2 \end{pmatrix}$, with no other rules to replace \vec{N} , while \vec{K} and \vec{M} are eligible for further operations.

The operations can be translated into grammar rules as follows (in each of the rules \vec{Z} respectively 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, 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 $\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} \rightarrow \begin{pmatrix} S \\ S \end{pmatrix}$ resp. $Y \rightarrow \bullet S$.

Finally, note that $(s \circ n)^\infty(C) = n^\infty(s^\infty(C))$ 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 C'' nested in the inner closure, we may only add T at positions that have not been eligible for nesting earlier.

Since we use $n''^\infty(C)$ only as a first closure we can easily leave out S at forbidden positions.

- **Stem extension:** A single stem extension can be achieved using $\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} \rightarrow \begin{pmatrix} O_1 I_1 \\ I_2 O_2 \end{pmatrix}$. As stem extending \vec{I} or \vec{O} will yield the same result, we forbid 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. The components of the other symbols can not occur immediately neighbored. For symbols \vec{Z} eligible for modified knot extension k'_i , we have to distinguish which of the resulting symbols inherits this property, requiring two rules $\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} \rightarrow \begin{pmatrix} O_1 I_1 \\ J_2 O_2 \end{pmatrix}$ and $\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} \rightarrow \begin{pmatrix} O_1 I_1 \\ I_2 Q_2 \end{pmatrix}$, where \vec{J} and \vec{Q} are eligible for further knot extensions. Additionally, \vec{J} is required to be extended at least once more and, as \vec{I} , disallowed from further stem extensions.

- **Knot extension:** The three cases of knot extension are modeled by the rules $\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} \rightarrow \begin{pmatrix} C_1 B_1 C_2 \\ B_2 \end{pmatrix}$, $\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} \rightarrow \begin{pmatrix} D_1 \\ E_1 D_2 E_2 \end{pmatrix}$, and $\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} \rightarrow \begin{pmatrix} L_1 R_1 \\ L_2 R_2 \end{pmatrix}$. 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 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 the first type.

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

As applying the third kind of knot extension to \vec{L} or \vec{R} is equivalent, we disallow the former.

In case of the modified knot extension k'_1 , the third rule is omitted and we have to additionally disallow \vec{C} and \vec{E} from knot extension.

For $A_1 A_2$, all types of knot extensions are equivalent. Thus we disallow \vec{A} from those and instead add $K_1 M_1 K_2 M_2$ to the base set. This structure also needs special treatment:

Figure 6 shows all possible results of extending \vec{M} , the white arc standing for \vec{K} , the gray and black arcs for the symbols derived from \vec{M} . However, we can also view the black arc as standing for \vec{M} and the white and gray arcs for symbols derived in one extension from \vec{K} . Thus to get an unambiguous grammar we disallow \vec{M} from both stem and knot extension.

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 $M_1 F_1 M_2 F_2$ 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 $G_1 H_1 G_2 H_2$ and disallow \vec{G} and \vec{H} from stem extensions of the second kind since these 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 being, which symbol is extended). Since these structures 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 preceded by a stem extension of the first kind or to the outside before terminating.

Now we can apply these rules to create unambiguous grammars for our classes. All grammars given below are simplified by eliminating symbols with only one associated rule or with all rules identical to another symbol.

PKF: Combining the rules for the base set A and nesting as first closure we get: $S \rightarrow \epsilon + \bullet S + [S]S$.

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). This results in:

$$S \rightarrow \epsilon + \bullet T + [T]T + TK_1^{(1)} K_1^{(2)} K_2^{(1)} K_2^{(2)}, T \rightarrow \epsilon + \bullet T + [T]T, \begin{pmatrix} K_1 \\ K_2 \end{pmatrix} \rightarrow \begin{pmatrix} K_1 [T] \\ [T] K_2 \end{pmatrix} + \begin{pmatrix} [T] \\ [T] \end{pmatrix}.$$

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

$$S \rightarrow \epsilon + \bullet T + [T]T + TK_1^{(1)} K_1^{(2)} K_2^{(1)} K_2^{(2)} + TK_1^{(1)} K_1^{(2)} K_1^{(3)} K_2^{(2)} K_2^{(3)} + TK_1^{(1)} K_1^{(2)} K_2^{(1)} K_1^{(3)} K_2^{(2)} K_2^{(3)} + TK_1^{(1)} K_1^{(2)} K_1^{(3)} K_2^{(2)} K_1^{(4)} K_2^{(3)} K_2^{(4)} K_2^{(1)}, T \rightarrow \epsilon + \bullet T + [T]T, \begin{pmatrix} K_1 \\ K_2 \end{pmatrix} \rightarrow \begin{pmatrix} K_1 [T] \\ [T] K_2 \end{pmatrix} + \begin{pmatrix} [T] \\ [T] \end{pmatrix}.$$

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

$$S \rightarrow \epsilon + \bullet S + [S]S + K_1^{(1)} SK_1^{(2)} SK_2^{(1)} SK_2^{(2)} S, \begin{pmatrix} K_1 \\ K_2 \end{pmatrix} \rightarrow \begin{pmatrix} K_1 [S] \\ [S] K_2 \end{pmatrix} + \begin{pmatrix} [S] \\ [S] \end{pmatrix}.$$

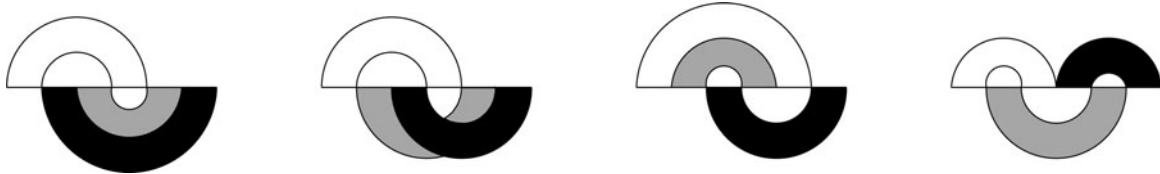


FIG. 6. Extending a basic pseudoknot.

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: $S \rightarrow \epsilon + \bullet S + [S]S + K_1^{(1)}SK_1^{(2)}TK_2^{(1)}SK_2^{(2)}S$, $T \rightarrow \epsilon + \bullet T$,
 $\begin{pmatrix} K_1 \\ K_2 \end{pmatrix} \rightarrow \begin{pmatrix} O_1 \\]TO_2 \end{pmatrix} + \begin{pmatrix} [\\] \end{pmatrix}$, $\begin{pmatrix} O_1 \\ O_2 \end{pmatrix} \rightarrow \begin{pmatrix} O_1[T \\]TO_2 \end{pmatrix} + \begin{pmatrix} [T \\] \end{pmatrix}$.

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: $S \rightarrow \epsilon + \bullet S + [S]S + K_1^{(1)}SK_1^{(2)}SK_2^{(1)}SK_2^{(2)}S$, $T \rightarrow \epsilon + \bullet T$,
 $\begin{pmatrix} K_1 \\ K_2 \end{pmatrix} \rightarrow \begin{pmatrix} O_1 \\]TO_2 \end{pmatrix} + \begin{pmatrix} [\\] \end{pmatrix}$, $\begin{pmatrix} O_1 \\ O_2 \end{pmatrix} \rightarrow \begin{pmatrix} O_1[T \\]TO_2 \end{pmatrix} + \begin{pmatrix} [T \\] \end{pmatrix}$.

D&P: Compared to R&G nesting is moved from initial closure to final closure yielding: $S \rightarrow \epsilon + \bullet S + [S]S + K_1^{(1)}K_1^{(2)}K_2^{(1)}K_2^{(2)}$, $\begin{pmatrix} K_1 \\ K_2 \end{pmatrix} \rightarrow \begin{pmatrix} K_1[S] \\]SK_2 \end{pmatrix} + \begin{pmatrix} [S] \\] \end{pmatrix}$.

A&U: Here the base class is changed. This gives (after simplifying): $S \rightarrow \epsilon + \bullet S + A_1A_2$,

$$\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \rightarrow \begin{pmatrix} M_1 \\ K_1M_2K_2 \end{pmatrix} + \begin{pmatrix} [S] \\] \end{pmatrix}, \begin{pmatrix} M_1 \\ M_2 \end{pmatrix} \rightarrow \begin{pmatrix} M_1K_1^{(1)} \\ K_2^{(1)}K_1^{(2)}M_2K_2^{(2)} \end{pmatrix} + \begin{pmatrix} K_1 \\ K_2 \end{pmatrix}, \begin{pmatrix} K_1 \\ K_2 \end{pmatrix} \rightarrow \begin{pmatrix} K_1[S] \\]SK_2 \end{pmatrix} + \begin{pmatrix} [S] \\] \end{pmatrix}.$$

R&N: By changing base class: $S \rightarrow \epsilon + \bullet S + [S]S + K_1^{(1)}K_1^{(2)}K_2^{(1)}K_2^{(2)} + K_1^{(1)}K_1^{(2)}K_2^{(1)}K_2^{(3)}K_2^{(2)}K_2^{(3)} + K_1^{(1)}K_1^{(2)}K_1^{(3)}K_2^{(1)}K_2^{(2)}K_2^{(3)}K_2^{(4)}$, $\begin{pmatrix} K_1 \\ K_2 \end{pmatrix} \rightarrow \begin{pmatrix} K_1[S] \\]SK_2 \end{pmatrix} + \begin{pmatrix} [S] \\] \end{pmatrix}$.

UMK: For this class we have to add the rules for $k'_1(A)$: $S \rightarrow \epsilon + \bullet S + [S]S + K_1[SK_2]S + [SF_1]SF_2$,

$$\begin{aligned} \begin{pmatrix} K_1 \\ K_2 \end{pmatrix} &\rightarrow \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \begin{pmatrix} K_1[S] \\]SK_2 \end{pmatrix} + \begin{pmatrix} [SK_1]S \\ K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ [SD_2]S \end{pmatrix} + \begin{pmatrix} [S] \\] \end{pmatrix}, \\ \begin{pmatrix} O_1 \\ O_2 \end{pmatrix} &\rightarrow \begin{pmatrix} O_1[S] \\]SO_2 \end{pmatrix} + \begin{pmatrix} [S] \\] \end{pmatrix}, \\ \begin{pmatrix} D_1 \\ D_2 \end{pmatrix} &\rightarrow \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \begin{pmatrix} K_1[S] \\]SK_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ [SD_2]S \end{pmatrix} + \begin{pmatrix} [S] \\] \end{pmatrix}, \\ \begin{pmatrix} J_1 \\ J_2 \end{pmatrix} &\rightarrow \begin{pmatrix} [SK_1]S \\ K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ [SD_2]S \end{pmatrix}, \\ \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} &\rightarrow \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \begin{pmatrix} Z_1[S] \\]SZ_2 \end{pmatrix} + \begin{pmatrix} [SF_1]S \\ F_2 \end{pmatrix}, \\ \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} &\rightarrow \begin{pmatrix} O_1J_1 \\ J_2O_2 \end{pmatrix} + \begin{pmatrix} Z_1[S] \\]SZ_2 \end{pmatrix} + \begin{pmatrix} [SZ_1]S \\ Z_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ [SD_2]S \end{pmatrix} \end{aligned}$$

CCJ: Adding the rules for $k'_2(k(A))$ yields: $S \rightarrow \epsilon + \bullet S + [S]S + K_1[SK_2]S + [SF_1]SF_2 + G_1H_1G_2H_2$,

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

R&E: For this class we have to replace k' by k , giving: $S \rightarrow \epsilon + \bullet S + [S]S + K_1[SK_2]S$,

$$\begin{aligned} \begin{pmatrix} K_1 \\ K_2 \end{pmatrix} &\rightarrow \begin{pmatrix} K_1I_1 \\ I_2K_2 \end{pmatrix} + \begin{pmatrix} I_1K_1I_2 \\ K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ I_1D_2I_2 \end{pmatrix} + \begin{pmatrix} L_1K_1 \\ L_2K_2 \end{pmatrix} + \begin{pmatrix} [S] \\] \end{pmatrix}, \\ \begin{pmatrix} I_1 \\ I_2 \end{pmatrix} &\rightarrow \begin{pmatrix} I_1K_1I_2 \\ K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ I_1D_2I_2 \end{pmatrix} + \begin{pmatrix} L_1K_1 \\ L_2K_2 \end{pmatrix} + \begin{pmatrix} [S] \\] \end{pmatrix}, \\ \begin{pmatrix} D_1 \\ D_2 \end{pmatrix} &\rightarrow \begin{pmatrix} K_1I_1 \\ I_2K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ I_1D_2I_2 \end{pmatrix} + \begin{pmatrix} L_1K_1 \\ L_2K_2 \end{pmatrix} + \begin{pmatrix} [S] \\] \end{pmatrix}, \\ \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} &\rightarrow \begin{pmatrix} K_1I_1 \\ I_2K_2 \end{pmatrix} + \begin{pmatrix} I_1K_1I_2 \\ K_2 \end{pmatrix} + \begin{pmatrix} D_1 \\ I_1D_2I_2 \end{pmatrix} + \begin{pmatrix} [S] \\] \end{pmatrix}. \end{aligned}$$

4.2. Enumeration

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

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 $f : \vec{X} \rightarrow \vec{\alpha}$ with
 - the direct successors of η from left to right are labeled by the symbols of α_i ;
 - 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) $X_{i_1}, X_{i_2}, \dots, X_{i_{d(\vec{X})}}$, 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 an MCFG G the degree of ambiguity $\deg(w)$ of $w \in \mathcal{L}(G)$ by the number of different derivation trees for w . By $\deg_{X_{i_1}, X_{i_2}, \dots, X_{i_{d(\vec{X})}}}(w)$ 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 $\deg_{\circ}^G(w)$ 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 $\sum_{w \in T^n} \deg_{X_{i_1}, X_{i_2}, \dots, X_{i_{d(\vec{X})}}}(w) = \sum_{w' \in T^n} \deg_{X_1, X_2, \dots, X_{d(\vec{X})}}(w')$.*

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 $X_{i_1}, X_{i_2}, \dots, X_{i_{d(\vec{X})}}$, we have $w_i \in T^*$ 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 $\deg_{X_{i_1}, X_{i_2}, \dots, X_{i_{d(\vec{X})}}}$ to $\deg_{X_1, X_2, \dots, X_{d(\vec{X})}}$, 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 $X_{i_1}, X_{i_2}, \dots, X_{i_{d(\vec{X})}}$ implies a derivation forest for $w' \in T^n$ rooted $X_1, X_2, \dots, X_{d(\vec{X})}$. 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 $I_m = \{\vec{X}^{<1>}, \dots, \vec{X}^{<k>}\}$. 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

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

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 various²

²Note 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.

$X^{<i,j>}$. Here, we assume $n_i \in \mathbb{N}_0$, $1 \leq i \leq |i(f)|$, and $\sum_{n_1+n_2+\dots+n_{|i(f)|}=n-t(f)} \dots = \delta_{n,t(f)}$, 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 $\mathcal{X}^{<j>}(z) := \sum_{n \geq 0} \mathbf{X}^{<j>}(n)z^n$:

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

by making use of series convolution. According to Kuich and Salomaa (1986), 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 by³

$$\bigcup_{1 \leq j \leq k} \bigcup_{f \in R_j} \left\{ X^{<j>} \rightarrow A_f, A_f \rightarrow a^{t(f)} \bigodot_{1 \leq i \leq |i(f)|} X^{<i_i>} \right\}.$$

Here a is an arbitrary terminal symbol and $I := \bigcup_{f \in P_m} \{A_f\} \cup \bigcup_{1 \leq j \leq k} X^{<j>}$. 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 an MCFG and a plain context-free encoding. This is the idea of bijective combinatorics used, e.g., in Saule et al. (2011), 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 major progress for cases where such a bijection is out of reach.

Corollary 4.3. *Let $G_m = (I, d, T, P, \vec{X}^{<1>})$, $I = \{\vec{X}^{<1>}, \dots, \vec{X}^{<k>}\}$, an unambiguous MCFG without ε -rules⁴ and SE the system of equations where for each $\vec{X}^{<i>} \in I$ the following variable and corresponding equation is introduced:*

$$X^{(i)}(z) = \sum_{\vec{a}: \vec{X}^{<i>} \rightarrow \vec{a} \in P} \prod_{1 \leq j \leq d(\vec{X}^{<i>})} h(\alpha_j), \quad (1)$$

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 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. 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., Kuich and Salomaa, 1986) 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). ■

Remark: The assumption of ε -freeness is used in Kuich and Salomaa (1986) 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

³We introduce the new intermediates A_f to be able to distinguish cases in which different productions f, f' from R_j introduce the same string $a^{t(f)} \bigodot_{1 \leq i \leq |i(f)|} X^{<i_i>}$.

⁴For MCFGs an ε -rule is given by a rule like $\vec{A} \rightarrow (\varepsilon, \dots, \varepsilon)^T$.

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 Saule et al. (2011)—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).

Applying Corollary 4.3 to the grammar for class R&E 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

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

At this point, we proceed along the lines of Hille (1962) in order to determine an expansion of $S(z)$ at its dominant singularity from the implicit representation given before. We first compute the set of critical points for $S(z)$. Among those, we select the one of smallest modulus that is connected to a singularity getting $d = 0.2010468602 \dots$. By substituting $S = 1.3605149159 \dots + Y$ and $z = 0.2010468602 \dots - Z$ 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 ; resubstituting while applying the \mathcal{O} -transfer method (Flajolet and Odlyzko, 1990) finally yields

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

where $d^{-1} = 4.9739647694 \dots$ holds.

If we now want the size of a structure to be given by its number of base pairs, we have to tweak the application of the corollary slightly: 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 in more detail.

TABLE 3. THE ASYMPTOTICAL NUMBER OF PSEUDOKNOT STRUCTURES OF SIZE n

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	$\mathcal{O}(n^6)$

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 are 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 motifs, our approach is rather generic in 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 article, which allow for deciding in linear time whether or not a pseudoknot (dot bracket word) belongs to a given class. We also offer a corresponding web service to the scientific community. 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 for more than 10 years—was solved; many more original enumerating results have also been proven. It is worth mentioning that this technique is not restricted to the domain of RNA pseudoknots but can be used 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 (Kato et al., 2006, contains similar ideas).

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

ACKNOWLEDGMENTS

This research has been supported in part by DFG (German Research Foundation) research grant NE 1379/3-1.

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

- Akutsu, T. 2000. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discr. Appl. Math.* 104, 45–62.
- Cannone, J., Subramanian, S., Schnare, M., et al. 2002. The comparative RNA web (CRW) site: An online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics* 3, 15.
- Cao, S., and Chen, S.-J. 2009. Predicting structures and stabilities for h-type pseudoknots with interhelix loops. *RNA* 15, 696–706.
- Chen, H.-L., Condon, A., and Jabbari, H. 2009. An $O(n^5)$ algorithm for MFE prediction of kissing hairpins and 4-chains in nucleic acids. *J. Comp. Biol.* 16, 803–815.
- Chen, S.J. 2008. RNA folding: conformational statistics, folding kinetics, and ion electrostatics. *Annu Rev Biophys* 37, 197–214.
- Condon, A., Davy, B., Rastegari, B., et al. 2004. Classifying RNA pseudoknotted structures. *Theor. Comp. Sci.* 320, 35–50.
- Dirks, R.M., and Pierce, N.A. 2003. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *J. Comput. Chem.* 24, 1664–1677.
- Doudna, J.A., and Cech, T.R. 2002. The chemical repertoire of natural ribozymes. *Nature* 418, 222–228.
- Flajolet, P., and Odlyzko, A. 1990. Singularity analysis of generating functions. *SIAM J. Disc. Math.* 3, 216–240.
- Giedroc, D.P., and Cornish, P.V. 2009. Frameshifting RNA pseudoknots: structure and mechanism. *Virus Res.* 139, 193–208.
- Harrison, M.A. 1978. *Introduction to Formal Language Theory*. Addison-Wesley Publishing Company, Boston.

- Hille, E. 1962. *Analytic Function Theory, Volume II*. Chelsea Publishing Company, Providence, RI.
- Kato, Y., Seki, H., and Kasami, T. 2006. RNA pseudoknotted structure prediction using stochastic multiple context-free grammar. *IPSJ Digital Courier* 2, 655–664.
- Kuich, W., and Salomaa, A. 1986. *Semirings, Automata, Languages*. Springer, London.
- Lyngsø, R.B., and Pedersen, C.N. 2000a. Pseudoknots in RNA secondary structures. In *Proceedings of the 4th Annual International Conference on Computational Molecular Biology (RECOMB)*, 201–209.
- Lyngsø, R.B., and Pedersen, C.N. 2000b. RNA pseudoknot prediction in energy-based models. *J. Comp. Biol.* 7, 409–427.
- Matsui, H., Sato, K., and Sakakibara, Y. 2005. Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures. *Bioinformatics* 21, 2611–2617.
- Metzler, D., and Nebel, M.E. 2008. Predicting RNA secondary structures with pseudoknots by MCMC sampling. *J. Math. Biol.* 56, 161–181.
- Namy, O., Moran, S.J., Stuart, D.I., et al. 2006. A mechanical explanation of RNA pseudoknot function in programmed ribosomal frameshifting. *Nature* 441, 244–247.
- Reeder, J., and Giegerich, R., 2004. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics* 5, 104.
- Reidys, C.M., Huang, F.W., Andersen, J.E. et al. 2011. Topology and prediction of RNA pseudoknots. *Bioinformatics*. 27, 1076–1085.
- Rivas, E., and Eddy, S.R. 1999. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol.* 285, 2053–2068.
- Rødland, E.A. 2006. Pseudoknots in RNA secondary structures: Representation, enumeration, and prevalence. *J. Comp. Biol.* 13, 1197–1213.
- Saule, C., Régnier, M., Steyaert, J.-M., et al. 2011. Counting RNA pseudoknotted structures. *Journal of Computational Biology* 18, 1339–1351.
- Seki, H., Matsumura, T., Fujii, M., et al. 1991. On multiple context free grammars. *Theor. Comp. Sci.* 88, 191–229.
- Staple, D.W., and Butcher, S.E., 2005. Pseudoknots: RNA structures with diverse functions. *PLoS Biol.* 3, e213.
- Taufer, M., Licon, A., Araiza, R., et al. 2009. PseudoBase⁺⁺: an extension of PseudoBase for easy searching, formatting and visualization of pseudoknots. *Nucleic Acids Res.* 37, D127–D135.
- Theimer, C.A., Blois, C.A., and Feigon, J. 2005. Structure of the human telomerase RNA pseudoknot reveals conserved tertiary interactions essential for function. *Mol. Cell* 17, 671–682.
- Uemura Y., Hasegawa, A., Kobayashi, S., et al. 1999. Tree adjoining grammars for RNA structure prediction. *Theor. Comp. Sci.* 210, 277–303.
- Wild, S. 2010. An earley-style parser for solving the rna-rna interaction problem [Bachelor thesis]. Kaiserslautern University of Technology, Kaiserslautern, Germany.
- Zuker, M., and Stiegler, P. 1981. Optimal computer folding of larger RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.* 9, 133–148.

Address correspondence to:
 Prof. Markus E. Nebel
 Computer Science Department
 University of Kaiserslautern
 Gottlieb Daimler Str. 48
 Kaiserslautern 67663
 Germany

E-mail: nebel@informatik.uni-kl.de