Nebel *et al. Algorithms for Molecular Biology* 2011, **6**:24
http://www.almob.org/content/6/1/24

ALGORITHMS FOR MOLECULAR BIOLOGY

RESEARCH Open Access

Random generation of RNA secondary structures according to native distributions

Markus E Nebel, Anika Scheid* and Frank Weinberg

* Correspondence: a_scheid@cs.uni-kl.de
Department of Computer Science, University of Kaiserslautern, Germany

Abstract

Background: Random biological sequences are a topic of great interest in genome analysis since, according to a powerful paradigm, they represent the *background noise* from which the actual biological information must differentiate. Accordingly, the generation of random sequences has been investigated for a long time. Similarly, random object of a more complicated structure like RNA molecules or proteins are of interest.

Results: In this article, we present a new general framework for deriving algorithms for the non-uniform random generation of combinatorial objects according to the encoding and probability distribution implied by a stochastic context-free grammar. Briefly, the framework extends on the well-known recursive method for (uniform) random generation and uses the popular framework of admissible specifications of combinatorial classes, introducing weighted combinatorial classes to allow for the non-uniform generation by means of unranking. This framework is used to derive an algorithm for the generation of RNA secondary structures of a given fixed size. We address the random generation of these structures according to a realistic distribution obtained from real-life data by using a very detailed context-free grammar (that models the class of RNA secondary structures by distinguishing between all known motifs in RNA structure). Compared to well-known sampling approaches used in several structure prediction tools (such as SFold) ours has two major advantages: Firstly, after a preprocessing step in time $\mathcal{O}(n^2)$ for the computation of all weighted class sizes needed, with our approach a set of m random secondary structures of a given structure size n can be computed in worst-case time complexity $\mathcal{O}(m \cdot n \cdot \log(n))$ while other algorithms typically have a runtime in $\mathcal{O}(m \cdot n^2)$. Secondly, our approach works with integer arithmetic only which is faster and saves us from all the discomforting details of using floating point arithmetic with logarithmized probabilities.

Conclusion: A number of experimental results shows that our random generation method produces realistic output, at least with respect to the appearance of the different structural motifs. The algorithm is available as a webservice at http://wwwagak.cs.uni-kl.de/NonUniRandGen and can be used for generating random secondary structures of any specified RNA type. A link to download an implementation of our method (in Wolfram Mathematica) can be found there, too.

Keywords: Random generation, stochastic context-free grammars, RNA secondary structures

Background and Introduction

The topic of random generation algorithms (also called *samplers*) has been widely studied by computer scientists. As stated in [1], it has been examined under different perspectives, including combinatorics, algorithmics (design and/or engineering), as well as probability theory, where two of the main motivations for random sampling are the testing of combinatorial properties of structures (e.g. conjectured structural properties, quantitative aspects), as well as the testing of properties of the corresponding algorithms (with respect to correctness and/or efficiency).

As considers software engineering, the so-called *random testing* approach is commonly used to test implementations of particular algorithms, as it is usually not feasible to consider all possible inputs and unknown which of these inputs are among the most interesting ones. In fact, this approach requires for the generation of random instances of program inputs that obey various sorts of syntactic and semantic constraints (where the random instances usually ought to be of a preliminarily fixed input size in order to be comparable to each other).

In the Bioinformatics area, algorithms for generating random biological sequences have been investigated for a long time (see e.g. [2,3]). As stated in [4], random sequences are a topic of great interest in genome analysis, since according to a powerful paradigm, they represent the *background noise* from which the actual biological information must differentiate. Thus, random generation of combinatorial objects can be used in this context for simulations studies in order to isolate signal (unexpected events) from noise (statistically unavoidable regularities). In fact, according to [4], random biological sequences are for instance widely used for the detection of over-represented and under-represented motifs, as well as for determining whether scores of pairwise alignments are relevant or not: although there exist analytic approaches for these kinds of problems, for the most complex cases, it is often still necessary to be able to alternatively use a corresponding experimental approach (based on randomly generated sequences obtained from a computer programm). For this purpose, random sequences must obviously obey to a certain model that takes into account some relevant properties of actual real-life sequences, where such models are usually based on statistical parameters only. However, it is known that these classical models can be enriched by adding structural parameters (see [4]). Over the past years, several methods have been proposed for the random generation of more complex structures, where special attention has been paid to *RNA secondary structures*. RNA is a single-stranded nucleotide polymer and a major component of cellular processes (like DNA and proteins). An RNA strand is formed by linking together certain nucleotide units. The specific sequence of nucleotides along this chain is called the *primary structure* of the molecule. By pairing of nucleotides that are not linked in this chain (i.e. by the so-called effects of base pairing), the linear primary structure is folded into a three-dimensional conformation, called the *tertiary structure*, which in many cases determines the function of the molecule. Most of the 3D structure is determined by the intramolecular base-pairing interactions *in the plane*, which together form the *secondary structure* of the molecule. For this reason, *pseudoknots* (induced by crossing base pairs) are considered as tertiary interactions and are usually not permitted in the definition of secondary structure. As unknotted structures contain only nested base pairs and are thus essentially two-dimensional, they can be modeled as planar graphs. This rather descriptive and commonly used planar graph model for RNA secondary structures was first formalized in [5]. An example is shown in Figure 1.

Most of the existing random generation algorithms for RNA secondary structures are used for predicting the structure of a given RNA sequence (see e.g. [6,7]), while others can be employed for instance for evaluating structure comparison softwares [8]. Note that secondary structure prediction methods based on random sampling represent a non-deterministic counterpart to the up-to-date most successful and popular physics-based prediction methods that make use of the energy minimization paradigm and are realized by dynamic programming algorithms (see e.g. [9-12]). Random sampling also differs from the stochastical RNA structure prediction approach that is based on context-free modeling of structural motifs and adding some statistical parameters observed in real-life data by assigning probabilities to the corresponding motifs (see e.g. [13-15]). Nevertheless, it should be mentioned that statistical sampling methods like [6,7] used for RNA structure prediction are based on thermodynamics and thus inevitably inherit the problems and imprecisions related to energy minimizing methods, which are caused by the still incomplete commonly used free energy models for RNAs. In order to overcome these pitfalls, one could take the competing point of view and consider only typical structural information observed in a set of sample data as the basis for a new random generation method. If that information draws a realistic picture for all the different motifs of a molecule's folding, the corresponding sampling method is likely to produce realistic results. Accordingly, several authors made use of stochastic context free grammars and employed machine-learning techniques to train parameter values from a set of known secondary structures. Such grammars have widely been used in a predictive mode (see, e.g., [14]) but there are also successful examples of applications where the random sampling of derivation trees has been the core of the method (see,

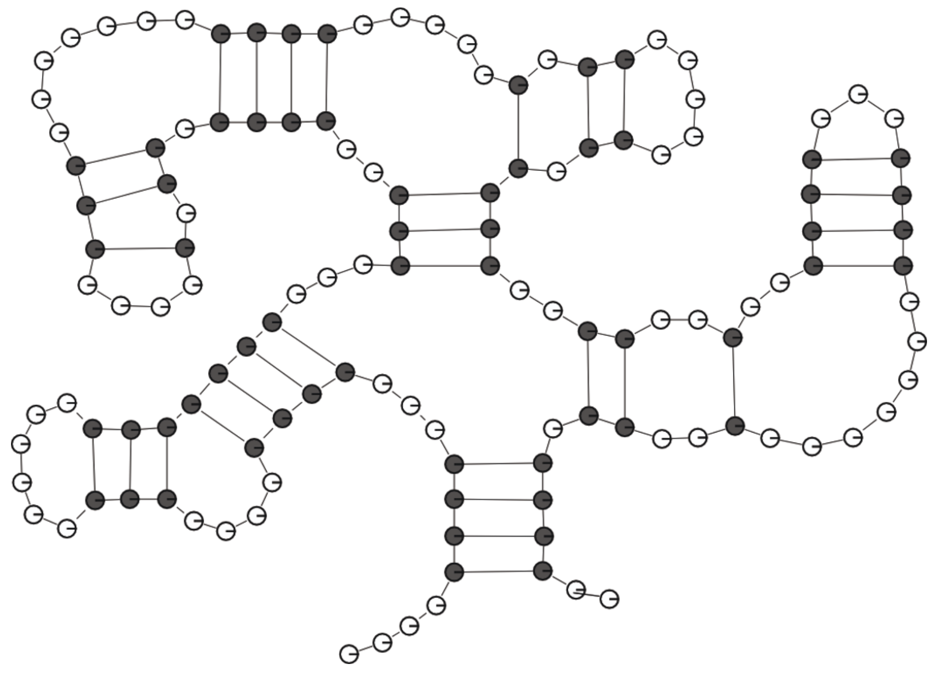
Figure 1 An RNA secondary structure. Unpaired and paired bases are represented by white and gray points, respectively.

e.g., [16-18] but also [19] for examples). In the present paper, we follow that line of ideas and rely on the approach of the technical report [20] to develop a new algorithm for the (non-uniform) random generation of RNA secondary structures (without pseudoknots) according to a distribution induced by a set of sample RNA data (note that the algorithm actually generates secondary structures for a preliminary fixed size, *not* for a given RNA sequence of this size, which means we take the combinatorial point of view and completely abstract from sequence).

The main contribution of this manuscript is the derivation of a new and efficient algorithm for the random generation of RNA secondary structures according to an elaborate and thus very realistic model. For this purpose we use and generalize the approach from [20]. Particularly, our random generation method is based on a sophisticated context-free grammar for unknotted structures which, in order to model the class of all considered RNA secondary structures as realistic as possible, distinguishes between all known structural motifs that may occur in unknotted RNA secondary structure. This means that any structural feature is modeled by one or more specific grammar rules with corresponding probabilities observed from real-life data. Note that this grammar is actually a special variant of the comprehensive grammar used in [21] for deriving a realistic RNA structure model and for performing the first ever analytical analysis of the expected free energy of a random secondary structure (of a specified RNA type). Actually, that grammar has been designed as a mirror of the famous Turner energy model [22,23] which serves as the foundation for most of the existing physics-based RNA structure prediction methods: all structural motifs for which there are different thermodynamic rules and parameters are created by distinct production rules (with corresponding probabilities).

According to [20], our sampling method involves a weighted *unranking* algorithm for obtaining the final structures. Briefly, considering an arbitrary structure class of size (cardinality) c, a corresponding unranking method uses a well-defined ordering of all class elements (according to a particular numbering scheme, the so-called *ranking* method) and for a given input number $r \in \{1,..., c\}$ outputs the structure with rank r in the considered ordering. That way, the random sampling based on a stochastic grammar - building heavily on the use of small floating point numbers - is translated into an unranking algorithm using integer values only. Notably, a complete structure of size n is generated by recursively unranking the distinct structural components from the corresponding subclasses (of substructures with sizes less than n). In our case, the weighted unranking algorithm requires a precomputation step in worst-case time $\mathcal{O}(n^2)$ for computing all weighted class sizes up to input size n. The worst-case complexity for generating a secondary structure of size n at random is then given by $\mathcal{O}(n \log n)$ since we are ranking structures according to the *boustrophedon order* (see e.g. [7]).

By the end of this paper, we analyze the quality of randomly generated structures by considering some experimental results. First, we will consider statistical indicators of many important parameters related to particular structural motifs and compare the ones observed in the used sample set of real world RNA data to those observed in a corresponding set of random structures. Their comparison measures indicate that our method actually generates realistic RNA structures. Obviously, an algorithm which, for a given structure size n, produces random RNA secondary structures that are - related to expected shapes of such structures -in most cases realistic is a major improvement over existing approaches which, for example, are only capable of generating secondary structures uniformly for size *n*. Furthermore, we will consider the two different free energy models defined in [21] for RNA secondary structure (with unknown RNA sequence) to get further evidence of the good quality of our random generation method (with respect to free energies and thus rather likely also with respect to appearance of the different structural motifs of RNA).

Prior Results and Basic Definitions

Uniform Random Generation

In the past, the problem of *uniform* random generation of combinatorial structures, that is the problem of randomly generating objects (of a preliminary fixed input size) of a specified class that have the same or similar properties, has been extensively studied. Special attention has been paid on the wide class of *decomposable structures* which are basically defined as combinatorial structures that can be constructed recursively in an unambiguous way.

In principle, two general (systematic) approaches have been developed for the uniform generation of these structures: First, the *recursive method* originated in [24] (to generate various data structures) and later systematized and extended in [25] (to decomposable data structures), where general combinatorial decompositions are used to generate objects at random based on counting possibilities. Second and more recently, the so-called *Boltzmann method* [1,26], where random objects (under the corresponding Boltzmann model) have a fluctuating size, but objects with the same size invariably occur with the same probability. Note that according to [26], Boltzmann samplers may be employed for approximate-size (objects with a randomly varying size are drawn) as well as fixed-size (objects of a strictly fixed size are drawn) random generation and are an alternative to standard combinatorial generators based on the recursive method. However, fixed-size generation is considered the standard paradigm for the random generation of combinatorial structures.

(Admissible) Constructions and Specifications

According to [25], a decomposable structure is a structure that admits an equivalent *combinatorial specification*:

Definition 0.1 ([25]). Let $\mathcal{A} = (\mathcal{A}_1, ..., \mathcal{A}_r)$ be an *r*-tuple of classes of combinatorial structures. A *specification* for $\mathcal{A}$ is a collection or *r* equations with the *i*th equation being of the form $\mathcal{A}_i = \phi_i(\mathcal{A}_1, ..., \mathcal{A}_r)$, where φ_i denotes a term built of the $\mathcal{A}_j$ using the *constructions* of disjoint union, cartesian product, sequence, set and cycle, as well as the initial (neutral and atomic) classes.

The needed formalities that will also be used in the sequel are given as follows:

Definition 0.2 ([27]). If $\mathcal{A}$ is a combinatorial class, then $\mathcal{A}^n$ denotes the class of objects in $\mathcal{A}$ that have size (defined as number of atoms) *n*. Furthermore:

- Objects of size 0 are called *neutral objects* or *tags* and a class consisting of a single neutral object ϵ is called a *neutral class*, which will be denoted by ε (ε_1, ε_2,... to distinguish multiple neutral classes containing the objects ϵ_1, ϵ_2, ..., respectively).

- Objects of size 1 are called *atomic objects* or *atoms* and a class consisting of a single atomic object is called an *atomic class*, which will be denoted by $\mathcal{Z}$($\mathcal{Z}_a$, $\mathcal{Z}_b$,... to distinguish the classes containing the atoms a,b,..., respectively).
- If $\mathcal{A}_1, ..., \mathcal{A}_k$ are combinatorial classes and ϵ_1, ..., ϵ_k are neutral objects, the *combinatorial sum* or *disjoint union* is defined as $\mathcal{A}_1 + ... + \mathcal{A}_k := (\varepsilon_1 \times \mathcal{A}_1) \cup ... \cup (\varepsilon_k \times \mathcal{A}_k)$ where ∪ denotes set theoretic union.
- If $\mathcal{A}$ and $\mathcal{B}$ are combinatorial classes, the *cartesian product* is defined as $\mathcal{A} \times \mathcal{B} := \left\{(\alpha, \beta)\,|\alpha \in \mathcal{A} \text{ and } \beta \in \mathcal{B}\right\}$, where size($\alpha$, β) = size(α) + size(β).

Note that the constructions of disjoint union, cartesian product, sequence, set and cycle are all admissible:

Definition 0.3 ([27]). Let ϕ be an m-ary construction that associates to a any collection of classes $\mathcal{B}_1, ..., \mathcal{B}_m$ a new class $\mathcal{A} := \phi\,[\mathcal{B}_1, ..., \mathcal{B}_m]$. The construction ϕ is *admissible* iff the counting sequence (a_n) of $\mathcal{A}$ only depends on the counting sequences ($b_{1,n}$),..., ($b_{m,n}$) of $\mathcal{B}_1, ..., \mathcal{B}_m$, where the counting sequence of a combinatorial class $\mathcal{A}$ is the sequence of integers $(a_n)_{n\geq 0}$ for $a_n = \text{card}\,(\mathcal{A}^n)$.

The framework of (admissible) specifications obviously resembles that of *context-free grammars (CFGs)* known from formal language theory (note that we assume the reader has basic knowledge of the notions concerning context-free languages and grammars. An introduction can be found for instance in [28]). In order to translate a CFG into the framework of admissible constructions, it is sufficient to make each terminal symbol an atom and to assume each non-terminal A to represent a class $\mathcal{A}$ (the set of all words which can be derived from non-terminal A). However, for representing CFGs, only the admissible constructions disjoint union, cartesian product and sequence are needed: Words are constructed as cartesian products of atoms, sentential forms as cartesian products of atoms and the classes assigned to the corresponding non-terminal symbols. For instance, a production rule $A \to aB$ translates into the symbolic equation $\mathcal{A} = a \times \mathcal{B}$. Different production rules with the same left-hand side give rise to the union of the corresponding cartesian products. Nevertheless, it should be noted that [25] also shows how to reduce specifications to *standard form*, where the corresponding standard specifications constitute the basis of the recursive method for uniform random generation and extends the usual *Chomsky normal form (CNF)* for CFGs. Briefly, in standard specifications, all sums and products are binary and the constructions of sequences, sets and cycles are actually replaced with other constructions (for details see [25]).

The prime advantage of standard specifications is that they translate directly into procedures for computing the sizes of all combinatorial subclasses of the considered class $\mathcal{C}$ of combinatorial objects. This means they can be used to count the number of structures of a given size that are generated from a given non-terminal symbol. Moreover, standard specifications immediately translate into procedures for generating one such structure uniformly at random. The corresponding procedures (for class size calculations and structure generations) are actually required for (uniform) random generation of words of a given CFG by means of unranking.

Simply speaking, the unranking of decomposable structures (like for instance RNA secondary structures which can be uniquely decomposed into distinct structural components) works as follows: Each structure s in the combinatorial class $\mathcal{S}^n$ of all feasible structures having size n is given a number (rank) $i \in \left\{0, ..., \text{card}\,(\mathcal{S}^n) - 1\right\}$, defined by a particular ranking method. Based on this ordering of the considered structure class $\mathcal{S}^n$, the corresponding unranking algorithm for a given input number $i \in \{0, ..., \text{card}(\mathcal{S}^n) - 1\}$ computes the single structure $s \in \mathcal{S}^n$ having number i in the ranking scheme defined for class $\mathcal{S}^n$.

Note that in this context of unranking particular elements from a considered structure class, the corresponding algorithms make heavy use of their decomposability, as the distinct structural components are unranked from the corresponding subclasses. In fact, the class sizes can be derived according to the following recursion:

$$\text{size}(\mathcal{C}, n) := \begin{cases} 1 & \mathcal{C} \text{ is neutral and } n = 0, \\ 0 & \mathcal{C} \text{ is neutral and } n \neq 0, \\ 1 & \mathcal{C} \text{ is atomic and } n = 1, \\ 0 & \mathcal{C} \text{ is atomic and } n \neq 1, \\ \sum_{i=1}^{k} \text{size}(\mathcal{A}_i, n) & \mathcal{C} = \mathcal{A}_1 + \ldots + \mathcal{A}_k, \\ \sum_{j=0}^{n} size(\mathcal{A}, j) \cdot \text{size}(\mathcal{B}, n-j) & \mathcal{C} = \mathcal{A} \times \mathcal{B}. \end{cases}$$

Note that when computing the sums for cartesian products, we can either consider the values for j in the *sequential* (also called *lexicographic*) order (1, 2, 3,..., n) or in the so-called *boustrophedon* order $\left(1, n, 2, n-1, ..., \lceil \frac{n}{2} \rceil\right)$. In either case, given a fix number of considered combinatorial (sub)classes (or corresponding non-terminal symbols), the precomputation of all class size tables up to size n requires $\mathcal{O}(n^2)$ operations on coefficients. One random generation step then needs $\mathcal{O}(n^2)$ arithmetic operations when using the *sequential method* and $\mathcal{O}\left(n \cdot \log(n)\right)$ operations when using the *boustrophedon method* (for details we refer to [25]). Obviously, using uniform unranking procedures to construct the ith structure of size n for a randomly drawn number i, any structure of size n is equiprobably generated. Consequently, in order to make sure that, for given size n and a sample set of random numbers i, the corresponding structures are in accordance with an appropriate probability distribution (as for instance observed from real-life RNA data), it is mandatory to use a corresponding non-uniform unranking method or an alternative non-uniform random generation approach.

Non-Uniform Random Generation

Coming back to the random testing problem from software engineering, we observe that generating objects of a given class of input data according to a uniform distribution is sufficient for testing the correctness of particular algorithms. However, if one intends to gather information about the "real-life behaviour" of the algorithm (e.g. with respect to runtime or space requirements), we need to perform simulations with input data that are as closely as possible related to corresponding application. This means to obtain suitable test data, we need to specify a distribution on the considered class that is similar to the one observed in real life and draw objects at random according to this (non-uniform) distribution. Deriving such a "realistic" distribution on a given class of objects can easily be done by modeling the class by an appropriate *stochastic context-free grammar (SCFG)*. Details will follow in the next section.

As regards RNA, it has been proven that both the combinatorial model (that is based on a uniform distribution such that all structures of a given size are equiprobable and that completely abstracts from the primary structure, see e.g. [29-31]) and the Bernoulli-model (which is capable of incorporating information on the possible RNA sequences for a given secondary structure, see e.g. [32-34]) for RNA secondary structures are rather unrealistic. However, modeling these structures by an appropriate SCFG yields a more realistic RNA model, where the probability distribution on all structures is determined from a database of real world RNA data (see e.g. [35,36]).

Based on this observation, the problem of *non-uniform* random generation of combinatorial structures has been recently addressed in [20]. There, it is described how to get algorithms for the random generation of objects of a previously fixed size according to an arbitrary (non-uniform) distribution implied by a given SCFG. In principle, the construction scheme introduced in [20] extends on the recursive method for the (uniform) random generation [25] and adapted it to the problem of unranking of [37]: the basic principle is that any (complex) combinatorial class can be decomposed into (or can be constructed from) simpler classes by using admissible constructions.

Essentially, in [20], a new admissible construction called *weighting* has been introduced in order to make non-uniform random generation possible. By weighting, we understand the generation of distinguishable copies of objects. Formally:

Definition 0.4. If $\mathcal{A}$ is a combinatorial class and λ is an integer, the *weighting* of $\mathcal{A}$ by λ is defined as $\lambda\mathcal{A} := \underbrace{\mathcal{A} + \ldots + A}_{\lambda \text{ times}}$. We will call two objects from a combinatorial class *copies of the same object* iff they only differ in the tags added by weighting operations.

For example, if we weight the class $\mathcal{A} = \{a\}$ by two, we assume the result to be the set $\{a, a\}$; weighting $\mathcal{B} = \{b\}$ by three generates $\{b,b,b\}$. Thus, $2\mathcal{A} + 3\mathcal{B} = \{a, a, b, b, b\}$ and within this class, a has relative frequency $\frac{2}{5}$, while b has relative frequency $\frac{3}{5}$. Hence, this way it becomes possible to regard non-uniformly distributed classes.

As weighting a class can be replaced by a disjoint union, $\text{size}(\lambda\mathcal{A}, n) = \lambda \cdot \text{size}(\mathcal{A}, n)$ and the complexity results from [37] also hold for weighted classes. Hence, the corresponding class size computations up to n need $\mathcal{O}(n^2)$ time.

Stochastic Context-Free Grammars

As already mentioned, *stochastic context-free grammars (SCFGs)* are a powerful tool for modeling combinatorial classes and the essence of the non-uniform random sampling approach that will be worked out in this article. Therefore, we will now give the needed background information.

Basic Concepts

Briefly, SCFGs are an extension of traditional CFGs: usual CFGs are only capable of modeling the class of all generated structures and thus inevitably induce a uniform distribution on the objects, while SCFGs additionally produce a (non-uniform) probability distribution on the considered class of objects. In fact, an SCFG is derived by equipping the productions of a corresponding CFG with probabilities such that the induced distribution on the generated language models as closely as possible the distribution of the sample data.

The needed formalities are given as follows:

Definition 0.5 ([38]). A *weighted context-free grammar (WCFG)* is a 5-tuple $\mathcal{G} = (I, T, R, S, W)$, where I (resp. T) is an alphabet (finite set) of intermediate (resp. terminal) symbols (I and T are disjoint), $S \in I$ is a distinguished intermediate symbol called *axiom,* $R \subset I \times (I \cup T)^*$ is a finite set of production rules and $W : R \to \mathbb{R}^+$ is a mapping such that each rule $f \in R$ is equipped with a weight $w_f := W(f)$. If $\mathcal{G}$ is a

WCFG, then $\mathcal{G}$ is a *stochastic context-free grammar (SCFG)* iff the following additional restrictions hold:

1. For all $f \in R$, we have $W(f) \in (0,1]$, which means the weights are probabilities.
2. The probabilities are chosen in such a way that for all $A \in I$, we have $\sum_{f \in R, Q(f)=A} w_f = 1$, where $Q(f)$ denotes the *premise* of the production f, i.e. the first component A of a production rule $(A, \alpha) \in R$. In the sequel, we will write $w_f : A \to \alpha$ instead of $f = (A, \alpha) \in R$, $w_f = w(f)$.

However, at this point, we decided to not recall the basic concepts regarding SCFGs, as they are not really necessary for the understanding of this article. The interested reader is referred to the corresponding section in [21]. For a more fundamental introduction on stochastic context-free languages, see for example [39]. In fact, the only information needed in the sequel is that if structures are modeled by a *consistent* SCFG, then the probability distribution on the production rules of the SCFG implies a probability distribution on the words of the generated language and thus on the modeled structures. To ensure that a SCFG gets consistent, one can for example assign relative frequencies to the productions, which are computed by counting the production rules used in the leftmost derivations of a finite sample of words from the generated language. For unambiguous SCFGs, the relative frequencies can actually be counted efficiently, as for every word, there is only one leftmost derivation to consider.

Modeling RNA Secondary Structure via SCFGs

Besides the popular planar graph representation of unknotted secondary structures, many other ways of formalizing RNA folding have been described in literature. One well-established example is the so called *bar-bracket representation*, where a secondary structure is modeled as a string over the alphabet $\Sigma := \{(,), |\}$, with a bar **|** and a pair of corresponding brackets ( ) representing an unpaired nucleotide and two paired bases in the molecule, respectively (see, e.g. [30]). Obviously, both models abstract from primary structure, as they only consider the number of base pairs and unpaired bases and their positions. Moreover, there exists a one-to-one correspondence between both representations, as illustrated by the following example:

Example 0.1. The secondary structure shown in Figure 1 has the following equivalent bar-bracket representation that can be decomposed into subwords corresponding to the basic structural motifs that are distinguished in state-of-the-art thermodynamic models:

$$\overbrace{||||((((\underbrace{(|||hel_1|||hel_2||hel_3|)}_{\text{multiloop(of degree 3)}}))))||}^{\text{exterior loop}}, \text{ where}$$

$$hel_1 = ((((\ \overbrace{||||\ (((\ \underbrace{||||||}_{\text{hairpin}}\ )))}^{\text{bulge left}}\ )))),\quad hel_3 = ((\ \overbrace{||\ (\ ||\underbrace{||\ ((((\ |||\ ))))\ |||||||}_{2\times 7\text{ interior loop}}\ )\ ||}^{2\times 2\text{ interior loop}}\ )),$$

$$\text{and } hel_2 = (((\ \overbrace{||\ ((((\ \ \underbrace{|hel_{2,1}|||||||}_{1\times 7\text{ interior loop}}\ \ ))))\ |||||hel_{2,2}}^{\text{multiloop(of degree 2)}}\ ))), \text{with}$$

$$hel_{2,1} = ((\ \overbrace{|\ (\ \underbrace{||||}_{\text{hairpin}}\ )}^{\text{single bulge left}}\ )) \text{ and } hel_{2,2} = (\ \overbrace{|((\ \underbrace{|||||}_{\text{hairpin}}\ ))|}^{1\times 1\text{ interior loop}}\ )$$

Note that the reading order of secondary structures is from left to right, which is due to the chemical structure of the molecule.

Consequently, secondary structures without pseudoknots can be encoded as words of a context-free language and the class of all feasible structures can thus effectively be modeled via a corresponding CFG. Basically, that CFG can be constructed to describe a number of classical constraints (e.g. the presence of particular motifs in structures) and it can also express long-range interactions (e.g. base pairings). By extending it to a corresponding SCFG, we can also model the fact that specific motifs of RNA secondary structures are more likely to be folded at certain stages than others (and not all possible motifs are equiprobable at any folding stage).

In fact, it is known for a long time that SCFGs can be used to model RNA secondary structures (see e.g. [40]). Additionally, SCFGs have already been used successfully for the prediction of RNA secondary structure [14,15]. Moreoever, they can be employed for identifying structural motifs as well as for deriving stochastic RNA models that are - with respect to the expected shapes - more realistic than other models [36]. Furthermore, note that an SCFG mirror of the famous Turner energy model has been used in [21] to perform the first analytical analysis of the free energy of RNA secondary structures; this SCFG marks a cornerstone between stochastic and pyhsics-based approaches towards RNA structure prediction.

Random Generation With SCFGs

SCFGs can easily be used for the random generation of combinatorial objects according to the probability distribution induced by a sample set, where the only problem is that they do not allow the user to fix the length of generated structures. In particular, given an SCFG $\mathcal{G}$ and the corresponding language (combinatorial class) $\mathcal{L}(\mathcal{G})$, a random word $w \in \mathcal{L}(\mathcal{G})$ can be generated in the following way:

- Start with the sentential form S (where S denotes the axiom of the grammar $\mathcal{G}$).
- While there are non-terminal symbols (in the currently considered sentential form), do the following:
  1) Let A denote the leftmost non-terminal symbol.
  2) Draw a random number r from the interval (0,1].
  3) Substitute symbol A by the right-hand side α of the production $A \rightarrow \alpha$ determined by the random number r.

This means consider all $m \geq 1$ rules $p_1 : A \rightarrow \alpha_1,..., p_m : A \rightarrow \alpha_m$ having left-hand side A, where according to the definition of SCFGs, $\sum_{i=1}^{m} p_i = 1$ must hold. Then, find $k \geq 1$ with $\sum_{i=1}^{k-1} p_i < r \leq \sum_{i=1}^{k} p_i$, i.e. determine $k \geq 1$ with $r \in \left(\sum_{i=1}^{k-1} p_i, \sum_{i=1}^{k} p_i\right]$. The production corresponding to the randomly drawn number $r \in (0,1]$ is then given by $A \rightarrow \alpha_k$ and hence, in the currently considered sentential form, the non-terminal symbol A is substituted by α_k.

- If there are no more non-terminal symbols, then the currently considered sentential form is equal to a word $w \in \mathcal{L}(\mathcal{G})$; w has been randomly generated.

Note that the choice of the production made in 3) according to the previously drawn random number is appropriate, since it is conform to the probability distribution on the grammar rules.

Example 0.2. Consider the language generated by the SCFG with productions ¾: $S \to \epsilon$ and ¼: $S \to (S)$. Thus, we start with the sentential form S, then consider the leftmost non-terminal symbol, which is given by S, and draw a random number $r \in (0,1]$. If $0 < r \leq$ ¾, the production determined by r is $S \to \epsilon$ and thus, we get the empty word and are finished. Otherwise, ¾ $< r \leq$ ¾ + ¼ = 1, which means we have to consider $A \to$ (S) for the substitution in step 3) and thus obtain the sentential form (S). Afterwards, we must repeat the process, as there is still one non-terminal symbol left.

Unfortunately, there is one major problem that comes with this approach for the (non-uniform) random generation of combinatorial objects: The underlying (consistent) SCFG $\mathcal{G}$ implies a probability distribution on the whole language $\mathcal{L}(\mathcal{G})$, such that we generate a word of arbitrary size. In order to fix the size, we can proceed along the following lines:

1) We translate the grammar $\mathcal{G}$ into a new framework which allows to consider fixed sizes for the random generation, such that
2) the distribution implied on $\mathcal{L}(\mathcal{G})$ conditioned on any fixed size n is kept within the new framework.

A well-known approach which allows for 1) is connected to the concept of admissible constructions used to describe a decomposable combinatorial class (see above). As the operations (like cartesian products, unions, and so on) used to construct the combinatorial objects are also used to define an order on them, it becomes possible to identify the ith object of a given size and the problem of generating objects uniformly at random reduces to the problem of unranking, that is the problem of constructing the object of order (rank) i, for i a random number (see e.g. [41]).

Remark. Some might think that with an appropriate SCFG (modeling a given class of objects) at hand, it is not really necessary to use an unranking method that implies cumbersome formalities such as admissible constructions and decomposable classes if we want to generate random objects of a fixed size n. As a matter of principle, they are right - we could also use a *conditional sampling* method: If we need to generate a word of size n from non-terminal symbol A, where there are $m \geq 1$ rules $f_i = A \to \alpha_i$, $1 \leq i \leq m$, having left-hand side A, then we just need to choose the next production f_i according to

$$\frac{Prob\left(A \to \alpha_i \Rightarrow^* x | size(x) = n\right)}{Prob\left(A \Rightarrow^* x | size(x) = n\right)},$$

which is the posterior probability that we used production rule f_i under the condition that a word of size n is generated.

Similarly, if the production rule is of the type $A \to BC$ (assuming the grammar is in Chomsky normal form (CNF), which does not pose a problem, as an unambiguous SCFG can be efficiently transformed into CNF [39]), we can choose a way to split size n into sizes j and $n - j$ for the lengths generated from non-terminal symbols B and C. This requires precomputing n *length-dependent* probabilities (i.e. all probabilities for generating a word of any length up to n) for each non-terminal symbol, which might seem to be similar (with respect to complexity) to precomputing all class sizes up to n for all considered combinatorial (sub)classes as needs to be done for unranking. However, there is a striking difference between the two approaches: While conditional sampling makes heavy use of rather small floating point values - with all the well-known problems and discomforting details like underflows or using logarithms associated with it - our unranking approach builds on integer values only which we assume a major advantage. There is another striking difference: length-dependent probabilities (which by the way yield a so-called length-dependent SCFG (LSCFG), see [42], and already have been used in [43]), require a very rich training set. In fact, if the RNA data set used for determining the distribution induced by the grammar is not rich enough, then the corresponding stochastic RNA model is underestimated and its quality decreases. This is especially a problem when considering comprehensive CFGs that distinguish between many different structural motifs in order to get a realistic picture of the molecules' behaviour; such a grammar should however be preferred over simple lightweight grammars as basis for a non-uniform random generation method. Nevertheless, this problem does not surface when sticking to conventional probabilities and the corresponding traditional SCFG model. Actually, since we consider a huge CFG where all possible structural motifs are created by distinct productions, we generally obtain realistic probability distributions and RNA models (see [21]).

Finally note that of course we could make use of random sampling strategies originally designed to sample structures connected to a given sequence in order to generate a random secondary structure only. However, such algorithms typically use a linear time to sample a single base pair (see, e.g., [6]) such that the time to sample a complete structure is quadratic in its length. This causes no problems for the original application of such algorithms since the sequence-dependent preprocessing which is part of their overall procedure is at least quadratic in time and thus the dominating part. Here our approach is of advantage (replacing a factor n by $log(n)$) and since our preprocessing only depends on the size of the structure to be generated it is performed once and stored to disk for later reuse. Last but not least we are not sure, if the different existing approaches just mentioned could easily be made as fast as ours by simple changes only.

Bottom line is that hooking up to unranking of combinatorial classes offers three significant benefit compared to conditional sampling, namely a fast sampling strategy, the usage of integers instead of floating point values and a greater independence of the richness of the training data (compared to length-dependent models). For this reason, we assume our unranking algorithm a valuable contribution, even though it requires a more cumbersome framework.

Unranking of Combinatorial Objects

The problem of unranking can easily be solved along the composition of the objects at hand, i.e. the operations used for its construction, once we know the number of possible choices for each substructure. Assume for example we want to unrank objects from a class $\mathcal{C} = \mathcal{A} + \mathcal{B}$. We will assume all elements of $\mathcal{A}$ to be of smaller order than those of $\mathcal{B}$ (this way we use the construction of the class to imply an ordering). Finding the ith element of $\mathcal{C}$, i.e. unranking class $\mathcal{C}$, now becomes possible by deciding whether

$\mathcal{A}$. In this case, we recursively call the unranking procedure for $\mathcal{A}$. Otherwise (i.e. if $i \geq \operatorname{card}(\mathcal{A})$), we consider $\mathcal{B}$, searching for its $(i - \operatorname{card}(\mathcal{A}))$)th element.

Formally, we first need to specify an order on all objects of the considered combinatorial class that have the same size. This can be done in a recursive way according to the admissible specification of the class:

Definition 0.6 ([37]). Neutral and atomic classes contain only one element, such that there is only one possible ordering. Furthermore, let $<_{\mathcal{C}^n}$ denote the ordering within the combinatorial class $\mathcal{C}^n$, then

- If $\mathcal{C} = \mathcal{A}_1 + \ldots + \mathcal{A}_k$ and $\gamma, \gamma' \in \mathcal{C}^n$, then $\gamma <_{\mathcal{C}^n} \gamma'$ iff

  $$\left[\gamma \in (\mathcal{A}_i)^n \text{ and } \gamma' \in (\mathcal{A}_j)^n \text{ and } i < j\right] \text{ or } \left[\gamma, \gamma' \in (\mathcal{A}_i)^n \text{ and } \gamma <_{(\mathcal{A}_i)^n} \gamma'\right].$$

- If $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ and $\gamma = (\alpha, \beta), \gamma' = (\alpha', \beta') \in \mathcal{C}^n$, then $\gamma <_{\mathcal{C}^n} \gamma'$ iff

  $$[\operatorname{size}(\alpha) < \operatorname{size}(\alpha')] \text{ or } \left[j = \operatorname{size}(\alpha) = \operatorname{size}(\alpha') \text{ and } \alpha <_{(\mathcal{A})^j} \alpha'\right] \text{ or } \left[\alpha = \alpha' \text{ and } \beta <_{(\mathcal{B})n-j} \beta'\right]$$

  when considering the lexicographic order (1, 2, 3,..., n), which is induced by the specification $\mathcal{C}^n = \mathcal{A}^0 \times \mathcal{B}^n + \mathcal{A}^1 \times \mathcal{B}^{n-1} + \mathcal{A}^2 \times \mathcal{B}^{n-2} + \ldots + \mathcal{A}^n \times \mathcal{B}^0$.

- If $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ and $\gamma = (\alpha, \beta), \gamma' = (\alpha', \beta') \in \mathcal{C}^n$, then $\gamma <_{\mathcal{C}^n} \gamma'$ iff

  $$\begin{aligned}&[\min(\operatorname{size}(\alpha), \operatorname{size}(\beta)) < \min(\operatorname{size}(\alpha'), \operatorname{size}(\beta'))] \text{ or}\\ &[\min(\operatorname{size}(\alpha), \operatorname{size}(\beta)) = \min(\operatorname{size}(\alpha'), \operatorname{size}(\mathcal{B}')) \text{ and } \operatorname{size}(\alpha) < \operatorname{size}(\alpha')] \text{ or}\\ &\left[j = \operatorname{size}(\alpha) = \operatorname{size}(\alpha') \text{ and } \alpha <_{(\mathcal{A})^j} \alpha'\right] \text{ or } \left[\alpha = \alpha' \text{ and } \beta <_{(\mathcal{B})^{n-j}} \beta'\right]\end{aligned}$$

  when considering the boustrophedon order $\left(1, n, 2, n-1, \ldots, \left\lceil \frac{n}{2} \right\rceil\right)$, induced by the specification $\mathcal{C}^n = \mathcal{A}^0 \times \mathcal{B}^n + \mathcal{A}^n \times \mathcal{B}^0 + \mathcal{A}^1 \times \mathcal{B}^{n-1} + \mathcal{A}^{n-1} \times \mathcal{B}^1 + \ldots$

Considering $<_{\mathcal{C}^n}$, the actual unranking algorithms are quite straightforward. Therefore, they will not be presented here and we refer to [20,44] for details.

Recall that in [20], the basic approach towards non-uniform random generation is weighting of combinatorial classes, as this makes it possible that the classes are non-uniformly distributed. If those combinatorial classes are to correspond to a considered SCFG, we have to face the problem that the maximum likelihood (ML) training introduces rational weights for the production rules while weighting as an admissible construction needs integer arguments.

When translating rational probabilities into integral weights, we have to assure that the relative weight of each (unambiguously) generated word remains unchanged. This can be reached by scaling all productions by the same factor (common denominator of all probabilities), while ensuring that derivations are of equal length for words of the same size (ensured by using grammars in CNF). However, a much more elegant way is to scale each production according to its contribution to the length of the word generated, that is, productions lengthening the word by k will be scaled by c^k. Since we consider CFGs, the lengthening of a production of the form $A \to \alpha$ is given by $|\alpha|$ - 1. However, this rule leads to productions with a conclusion of length 1 not being reweighted, hence we have to assure that all those productions already have integral weights. Furthermore, ϵ-productions need a special treatment. We don't want to discuss full details here and conclude by noticing that the *reweighting normal form (RNF)* keeps track of all possible issues:

Definition 0.7 ([20]). If $\mathcal{G} = (I, T, R, S, W)$ is a WCFG, $\mathcal{G}$ is said to be in *reweighting normal form (RNF)* iff

1. $\mathcal{G}$ is loop-free and ϵ-free.
2. For all $A \to \alpha \in R$ with $A = S$, we have $|\alpha| \le 1$.
3. For all $A \to \alpha \in R$ with $A \neq S$, we have $|\alpha| > 1$ or $W(A \to \alpha) \in \mathbb{N}$.
4. For *all* $A \in I$ there exists $\alpha \in (I \cup T)^*$ such that $A \to \alpha \in R$.

Note that the last condition (that any intermediate symbol occurs as premise of at least one production) is not required for reweighting, but necessary for the translation of a grammar into an admissible specification.

Definition 0.8 ([20]). A WCFG $\mathcal{G}$ is called *loop-free* iff there exists no nonempty derivation $A \Rightarrow^+ A$ for $A \in I$. It is called *ϵ-free* iff there exists no $(A, \epsilon) \in R$ with $A = S$ and there exists no $(A, \alpha_1 S \alpha_2) \in R$, where ϵ denotes the empty word.

If $\mathcal{G}$ and $\mathcal{G}'$ are WCFGs, then $\mathcal{G}$ and $\mathcal{G}'$ are said to be *word-equivalent* iff $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$ and for each word $w \in \mathcal{L}(\mathcal{G})$, we have $W(w) = W'(w)$.

In [20], it is shown how to transform an arbitrary WCFG to a word-equivalent, loop-free and ϵ-free grammar, that grammar to one in RNF and the latter to the corresponding admissible specification. Formally:

Theorem 0.1 ([39]). *If $\mathcal{G}$is a SCFG, there exists a SCFG $\mathcal{G}'$in Chomsky normal form (CNF) that is word-equivalent to $\mathcal{G}$, and $\mathcal{G}'$can be effectively constructed from $\mathcal{G}$.*

The construction given in [39] assumes that $\mathcal{G}$ is ϵ-free. It can however be extended to non-ϵ-free grammars by adding an additional step after the intermediate grammar $\mathcal{G}$ has been created (see e.g. [20]). Furthermore, it should be noted that an unambiguous grammar is inevitably loop-free.

Theorem 0.2 ([20]). *If $\mathcal{G}$is a loop-free, ϵ-free WCFG, there exists a WCFG $\mathcal{G}'$in RNF that is word-equivalent to $\mathcal{G}$and $\mathcal{G}'$can be effectively constructed from $\mathcal{G}$.*

Altogether, starting with an arbitrary unambiguous SCFG $\mathcal{G}_0$ that models the class of objects to be randomly generated, we have to proceed along the following lines:

- Transform $\mathcal{G}_0$ to a corresponding ϵ-free and loop-free SCFG $\mathcal{G}_1$.
- Transform $\mathcal{G}_1$ into $\mathcal{G}_2$ in RNF (where all production weights are rational).
- Reweight the production rules of $\mathcal{G}_2$ (such that all production weights are integral), yielding reweighted WCFG $\mathcal{G}_3$.
- Transform $\mathcal{G}_3$ (with integral weights) into the corresponding admissible specification.
- This specification (with weighted classes) can be translated directly
  - into a recursion for the function `size` of all involved combinatorial (sub) classes (where class sizes are weighted) and
  - into generating algorithms for the specified (weighted) classes,

yielding the desired weighted unranking algorithm for generating random elements of $\mathcal{L}(\mathcal{G}_0)$.

A small example that shows how to proceed from SCFG to reweighted normal form and the corresponding weighted combinatorial classes which allow for non-uniform generation by means of unranking is discussed in the Appendix.

Generating Random RNA Secondary Structures

We will now consider the previously discussed approach to construct a weighted unranking algorithm that generates random RNA secondary structures of a given size according to a realistic probability distribution. As for this paper, the corresponding probability distribution will be induced by a set of sample (SSU and LSU r)RNA secondary structures from the databases [45,46], which will be referred to as *biological database* in the sequel. However, the presented algorithm can easily be used for any other distribution, which can be defined by a database of known RNA structures of a particular RNA type; our webservice implementation accessible at http://wwwagak.cs.uni-kl.de/NonUniRandGen is actually able to sample random secondary structures of any specified RNA type. A link to download an implementation of our algorithm (in Wolfram Mathematica) can be found there, too.

Considered Combinatorial Class

According to the common definition of RNA secondary structure, we decided to consider the combinatorial class of all RNA secondary structures without pseudoknots that meet the stereochemical constraint of hairpin loops consisting of at least 3 unpaired nucleotides, formally:

Definition 0.9 ([21]). The language $\mathcal{L}$ containing exactly all RNA secondary structures is given by (note that according to this definition, completely unpaired structures are prohibited) $\mathcal{L} := \mathcal{L}_u\mathcal{L}_{lu}^{+}$, where $\mathcal{L}_{lu} := (\mathcal{L}_l)\mathcal{L}_u, \mathcal{L}_u := \{|\}^*$ is the language of all bar-bracket representations of single-stranded regions and $\mathcal{L}_l$ is the language of all bar-bracket representations of other possible substructures, i.e. is the smallest language satisfying the following conditions:

1. $\{|\}^+ \backslash \{|, ||\} \subset \mathcal{L}_l$ (bar-bracket representations of *hairpin loops*).
2. If $w \in \mathcal{L}_l$, then $(w) \in \mathcal{L}_l$ (bar-bracket representation of a *stacked pair*).
3. If $w \in \mathcal{L}_l$, then $\{|\}^+(w) \subset \mathcal{L}_l$ and $(w)\{|\}^+ \subset \mathcal{L}_l$ (bar-bracket representations of *bulge loops*).
4. If $w \in \mathcal{L}_l$, then $\{|\}^+(w)\{|\}^+ \subset \mathcal{L}_l$ (bar-bracket representations of *interior loops*).
5. If $w_1, \ldots, w_n \in \mathcal{L}_l$ and $n \geq 2$, then $\mathcal{L}_u(w_1)\mathcal{L}_u(w_2)\cdots\mathcal{L}_u(w_n)\mathcal{L}_u \subset \mathcal{L}_l$ (bar-bracket representations of *multibranched loops*).

The desired weighted unranking algorithm thus generates, for a given size n and a given number $i \in \{0, \ldots, \text{card}(\mathcal{L}^n) - 1\}$, the ith secondary structure $s \in \mathcal{L}^n$, where $\text{card}(\mathcal{L}^n) = \text{size}(\mathcal{L}, n)$ is the number of elements in the weighted class $\mathcal{L}^n$.

Considered SCFG Model

First, we have to find a suitable SCFG that generates $\mathcal{L}$ and models the distribution of the sample data as closely as possible. To reach this goal, it is important to appropriately specify the set of production rules in order to guarantee that all substructures that have to be distinguished are generated by different rules. This is due to the fact that by using only one production rule f to generate different substructures (e.g. any unpaired nucleotides independent of the type of loop they belong to), there is only one weight (the probability p_f of this production f) with which any of these substructures is generated, whereas the use of different rules $f_1,\dots, f_k$ to distinguish between these substructures implies that they may be generated with different probabilities $p_{f_1}, \dots p_{f_k}$, where $p_{f_1} + \dots + p_{f_k} = p_f$. This way, we ensure that more common substructures are generated with higher probabilities than less common ones.

Example 0.3. A (rather simple) unambiguous SCFG $\mathcal{G}_s$ generating the language $\mathcal{L}$ is given by:

$$\begin{array}{ll} w_1{:}S_s \to CA, & \\ w_2{:}A \to (B)C, & w_3{:}A \to (B)CA, \\ w_4{:}B \to |||C, & w_5{:}B \to CA, \\ w_6{:}C \to \epsilon, & w_7{:}C \to |C. \end{array}$$

This grammar unambiguously generates $\mathcal{L}$ for the following reasons:

- Every sentential form $C(B)C(B) \dots (B)C$ obviously is generated in a unique way; this resembles $\mathcal{L} = \mathcal{L}_u\mathcal{L}_{lu}^+$ and $\mathcal{L}_{lu} := (\mathcal{L}_l)\mathcal{L}_u$ of $\mathcal{L}$'s definition. The number of outermost pairs of brackets in the entire string uniquely determines the corresponding sentential form to be used.
- Now B either generates a hairpin-loop $|^{\geq 3}$, which unambiguously is possible by rules $B \to |||C$, $C \to |C$ and $C \to \epsilon$, or
- B itself has to generate at least one additional pair of brackets. In this case, $B \to CA$ must be applied (only A can generate brackets) and then $A \to (B)C$ resp. $A \to (B)CA$ are used; the number of outermost brackets to be generated (from B under consideration) again uniquely determines that part of the derivation.

When changing the production w_5: $B \to CA$ used to generate any possible k-loop for $k \geq 2$ (any loop that is not a hairpin loop) with probability w_5 into the two rules

$$w_{5.1} : B \to C(B)C, \quad w_{5.2} : B \to C(B)CA,$$

where $w_{5.1} + w_{5.2} = w_5$, it becomes possible to generate any possible 2-loop (i.e. a stacked pair, a bulge (on the left or on the right), or an interior loop) and all kinds of multiloops (i.e. any k-loop with $k \geq 3$) with different probabilities, which could increase the accuracy of the SCFG model. By additionally replacing the first of these two new rules, $w_{5.1} : B \to C(B)C$, by the four productions

$$w_{5.1.1} : B \to (B), \;\; w_{5.1.2} : B \to |C(B), \;\; w_{5.1.3} : B \to (B)C|, \;\; w_{5.1.4} : B \to |C(B)C|,$$

where $(w_{5.1.1} + \dots + w_{5.1.4}) + w_{5.2} = w_{5.1} + w_{5.2} = w_5$, we can distinguish between the different types of 2-loops more accurately, yielding a more realistic secondary structure model. In fact, in the case of significant differences of the new probabilities ($w_{5.1.1}$, ..., $w_{5.1.4}$ and $w_{5.2}$), we can expect a huge improvement in the model's accuracy. Note that it is not hard to see that changes to a grammar like the ones just discussed do not change the language generated. However, this is not at all obvious with respect to ambiguity of the grammar.

According to the previously mentioned facts (and the corresponding illustrations by Example 0.3), we decided that the basis for our weighted unranking algorithm should be the following e-free, loop-free and unambiguous (note that these are exactly the preliminary required conditions for the basis SCFG according to [20]) SCFG, which has been derived from the sophisticated SCFG presented in [21] that distinguishes between all known structural motifs that can be found in RNA secondary structure:

Definition 0.10. The unambiguous ϵ-free SCFG $\hat{\mathcal{G}}_{sto}$ generating exactly the language $\mathcal{L}$ is given by $\hat{\mathcal{G}}_{\text{sto}} = \left(I_{\hat{\mathcal{G}}_{\text{sto}}}, \sum_{\hat{\mathcal{G}}_{\text{sto}}}, R_{\hat{\mathcal{G}}_{\text{sto}}}, S'\right)$, where

$$I_{\hat{\mathcal{G}}_{sto}} = \{S', E, S, T, C, A, L, G, D, B, F, H, P, Q, R, V, W, O, J, K, M, X, Y, Z, N, U\},$$

$\sum_{\hat{\mathcal{G}}_{\text{sto}}} = \{(,), |\}$ and $R_{\hat{\mathcal{G}}_{\text{sto}}}$ contains exactly the following rules:

$$\left.\begin{array}{ll} \hat{p}_1 : S' \to E, & \\ \hat{p}_2 : E \to S, & \hat{p}_3 : E \to SC, \\ \hat{p}_4 : S \to A, & \hat{p}_5 : S \to TA, \\ \hat{p}_6 : T \to E, & \hat{p}_7 : T \to C, \end{array}\right\} \text{shape of exterior loop}$$

$$\hat{p}_8 : C \to \;—, \quad \hat{p}_9 : C \to C|, \;\rightsquigarrow \text{strands in exterior loop}$$

$$\hat{p}_{10} : A \to (L), \;\rightsquigarrow \text{initiate helix}$$

$$\hat{p}_{11} : L \to A, \;\hat{p}_{12} : L \to M, \;\rightsquigarrow \text{initiate stacked pair or multiple loop}$$

$$\hat{p}_{13} : L \to P, \;\hat{p}_{14} : L \to Q, \;\hat{p}_{15} : L \to R, \;\rightsquigarrow \text{initiate interior loop}$$

$$\hat{p}_{16} : L \to F, \;\hat{p}_{17} : L \to G, \;\rightsquigarrow \text{initiate hairpin loop or bulge loop}$$

$$\hat{p}_{18} : G \to A—, \;\hat{p}_{19} : G \to AD, \;\hat{p}_{20} : G \to \;—A, \;\hat{p}_{21} : G \to DA, \;\rightsquigarrow \text{shape of bulge loop}$$

$$\left.\begin{array}{ll} \hat{p}_{22} : D \to B|, & \\ \hat{p}_{23} : B \to \;—, & \hat{p}_{24} : B \to B—, \end{array}\right\} \text{strands in bulge loop}$$

$$\left.\begin{array}{l} \hat{p}_{25} : F \to \;—|—, \;\hat{p}_{26} : F \to \;—————, \;\hat{p}_{27} : F \to \;—————H, \\ \hat{p}_{28} : H \to \;—, \;\hat{p}_{29} : H \to H—, \end{array}\right\} \text{hairpin loop}$$

$$\hat{p}_{30} : P \to \;—A—, \;\hat{p}_{31} : P \to \;|A|—, \;\hat{p}_{32} : P \to \;—|A|, \;\hat{p}_{33} : P \to \;—|A——, \;\rightsquigarrow \text{small interior loops}$$

$$\left.\begin{array}{l} \hat{p}_{34} : Q \to \;—|O——, \;\hat{p}_{35} : Q \to \;—|V|, \\ \hat{p}_{36} : R \to \;|O——, \;\hat{p}_{37} : R \to \;——W|, \\ \hat{p}_{38} : V \to JO, \\ \hat{p}_{39} : W \to JA, \\ \hat{p}_{40} : O \to AK, \end{array}\right\} \text{other interior loops}$$

$$\left.\begin{array}{l} \hat{p}_{41} : J \to \;—, \;\hat{p}_{42} : J \to J—, \\ \hat{p}_{43} : K \to \;—, \;\hat{p}_{44} : K \to K—, \end{array}\right\} \text{strands in interior loop}$$

$$\left.\begin{array}{l} \hat{p}_{46} : M \to XY, \\ \hat{p}_{46} : X \to A, \;\hat{p}_{47} : X \to UA, \\ \hat{p}_{48} : Y \to Z, \\ \hat{p}_{49} : Z \to X, \;\hat{p}_{50} : Z \to XN, \\ \hat{p}_{51} : N \to Z, \;\hat{p}_{52} : N \to U, \end{array}\right\} \text{multiple loop}$$

$$\hat{p}_{53} : U \to \;—, \;\hat{p}_{54} : U \to U—, \;\rightsquigarrow \text{strands in multiple loop}$$

Figures 2 and 3 illustrate by examples how (parts of) secondary structures are generated by this SCFG, where we used $\triangleq$ to denote the full parse tree for $I \Rightarrow^* x$ (i.e. for consecutive applications of an arbitrary number of production rules that generate the subword x from the intermediate symbol I) in oder to obtain a more compact tree representation. In fact, it is easy to see that the overall structure is always produced by starting with the axiom S', while any particular substructure or structural motif that belongs to the combinatorial (sub)class $\mathcal{I}$ is created from the corresponding intermediate symbol I.

For our application it is crucial that $\hat{\mathcal{G}}_{sto}$ - as claimed its definition - is unambiguous. To prove this, we first note that $\hat{\mathcal{G}}_{sto}$ has been constructed starting from a simple grammar which generates $\mathcal{L}$ by iteratively replacing one production by several ones (like we

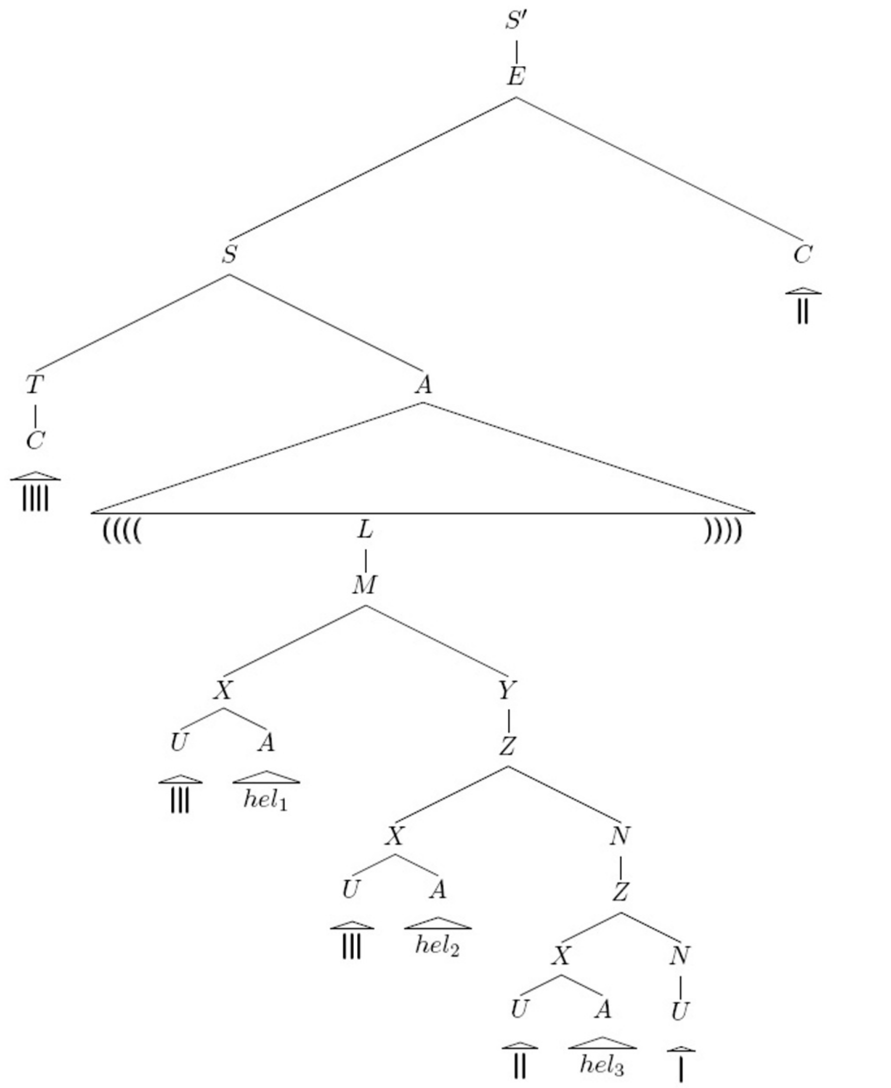

Figure 2 Unique parse tree for the bar-bracket word considered in Example 0.1 that corresponds to the planar secondary structure from Figure 1.

did in the previous example) in order to distinguish more and more structural motifs but without changing the language generated. Furthermore, a standard construction to make the grammar ϵ-free has been applied. That way, we can be sure that $\hat{\mathcal{G}}_{sto}$ generates $\mathcal{L}$ (formally this fact easily follows by obvious bi-simulation proofs for each substitution and by the proven correctness of the used construction to ensure ϵ-freeness). To prove unambiguity, we translate $\hat{\mathcal{G}}_{sto}$ into a system of equations for its structure generating function (see [47] for details) $S[z] = \sum_{w \in \mathcal{L}} d(w) z^{|w|}$ where $d(w)$ denotes the number of derivation trees $\hat{\mathcal{G}}_{sto}$ offers for w. Eliminating all but the variable associated with the axiom and simplifying (for this step we made use of Mathematica) yields the single equation

$$-z^5 + S[z](-1+z)(-1+z(2-S[z](-1+z)z+z^4)) = 0$$

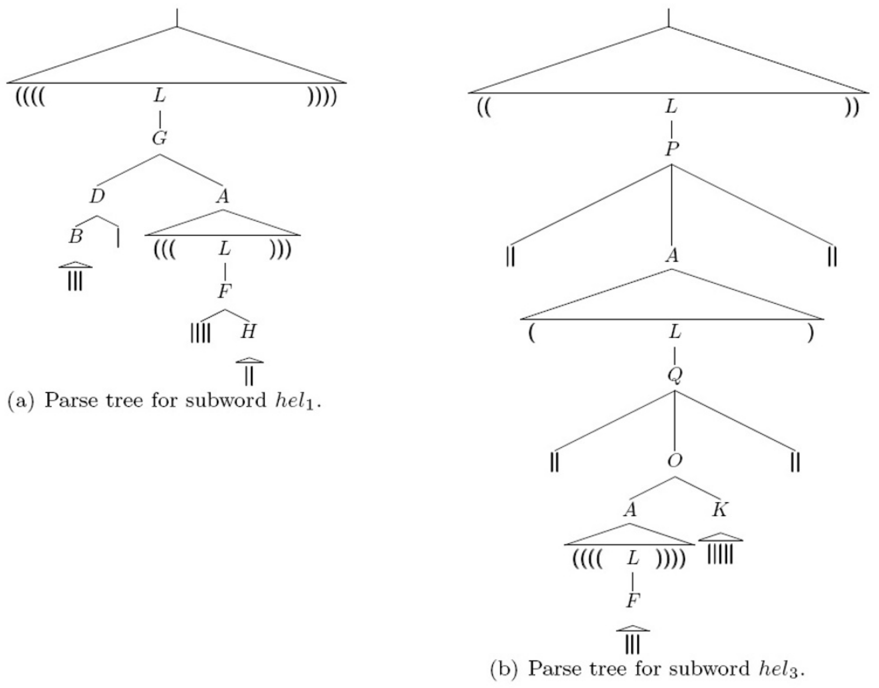
(a) Parse tree for subword hel_1.
(b) Parse tree for subword hel_3.

Figure 3 Particular subtrees of the tree presented in Figure 2.

This equation is exactly the one of our grammar $\mathcal{G}_s$ from Example 0.3 which proves that for all n both grammars have the same number of derivation trees for words of size n. Knowing that both grammars generate $\mathcal{L}$ and that $\mathcal{G}_s$ is unambiguous, the same can be concluded for $\hat{\mathcal{G}}_{sto}$.

Note that $\hat{\mathcal{G}}_{sto}$ contains more production rules (and more different non-terminal symbols) than the SCFG considered in [21], but this new grammar is ϵ-free and additionally, the right-hand side of every single production contains at most two non-terminal symbols, such that the resulting unranking algorithm has to consider less cases (i.e. less "else if ( )" cases). For details, see [20] and the Appendix.

Furthermore, it should be mentioned that we decided to assign relative frequencies to the production rules of $\hat{\mathcal{G}}_{sto}$, since such probabilities can be computed efficiently for unambiguous SCFGs. Moreover, by estimating the probabilities $\hat{p}_i$, $1 \leq i \leq 54$, by their relative frequencies, the resulting grammar $\hat{\mathcal{G}}_{sto}$ has the consistency property, which means $\hat{\mathcal{G}}_{sto}$ provides a probability distribution on the language $\mathcal{L}(\hat{\mathcal{G}}_{\text{sto}}) = \mathcal{L}$. In particular, it is well-known that relative frequencies in our context yield a maximum likelihood (ML) estimator for the rule probabilities and thus a consistent estimator for the parameter set. We have trained the probabilities (relative frequencies) of $\hat{\mathcal{G}}_{sto}$ from the structures $s \in \mathcal{L}\left(\hat{\mathcal{G}}_{\text{sto}}\right)$ given in our biological database. The resulting probabilities are given in Table 1, their floating point approximations, rounded to the third decimal place in Table 2.

In oder to see if over-fitting is an issue for our sophisticated grammar and its rich parameter set, i.e. to see if our training set is large enough to derive reliable values for the rule probabilities, we performed the following experiments: We selected a random

Table 1 The probabilities (relative frequencies) for the production rules of the SCFG $\hat{\mathcal{G}}_{sto}$ obtained by training it using our biological database

Nonterminal *Nt*	Probabilities of Rules with Premise *Nt*
S′	$\hat{p}_1 := 1$
E	$\hat{p}_2 := \frac{137}{6476}, \quad \hat{p}_3 := \frac{6339}{6476},$
S	$\hat{p}_4 := \frac{177}{12952}, \quad \hat{p}_5 := \frac{12775}{12952},$
T	$\hat{p}_6 := \frac{11086}{12775}, \quad \hat{p}_7 := \frac{1689}{12775},$
C	$\hat{p}_8 := \frac{14367}{148978}, \quad \hat{p}_9 := \frac{134611}{148978},$
A	$\hat{p}_{10} := 1,$
L	$\hat{p}_{11} := \frac{605069}{792975}, \hat{p}_{12} := \frac{31912}{792975}, \hat{p}_{13} := \frac{4912}{264325}, \hat{p}_{14} := \frac{5821}{158595},$ $\hat{p}_{15} := \frac{1893}{264325}, \hat{p}_{16} := \frac{2723}{31719}, \hat{p}_{17} := \frac{38399}{792975},$
G	$\hat{p}_{18} := \frac{11667}{38399}, \quad \hat{p}_{19} := \frac{7235}{38399}, \quad \hat{p}_{20} := \frac{11831}{38399}, \quad \hat{p}_{21} := \frac{7666}{38399},$
D	$\hat{p}_{22} := 1,$
B	$\hat{p}_{23} := \frac{4967}{12748}, \quad \hat{p}_{24} := \frac{7781}{12748},$
F	$\hat{p}_{25} := \frac{3912}{68075}, \quad \hat{p}_{26} := \frac{23208}{68075}, \quad \hat{p}_{27} := \frac{8191}{13615},$
H	$\hat{p}_{28} := \frac{8191}{40700}, \quad \hat{p}_{29} := \frac{32509}{40700},$
P	$\hat{p}_{30} := \frac{533}{4912}, \quad \hat{p}_{31} := \frac{1053}{4912}, \quad \hat{p}_{32} := \frac{2963}{14736}, \quad \hat{p}_{33} := \frac{7015}{14736},$
Q	$\hat{p}_{34} := \frac{4986}{29105}, \quad \hat{p}_{35} := \frac{24119}{29105},$
R	$\hat{p}_{36} := \frac{2357}{5679}, \quad \hat{p}_{37} := \frac{3322}{5679},$
V	$\hat{p}_{38} := 1,$
W	$\hat{p}_{39} := 1,$
O	$\hat{p}_{40} := 1,$
J	$\hat{p}_{41} := \frac{27441}{84620}, \quad \hat{p}_{42} := \frac{57179}{84620},$
K	$\hat{p}_{43} := \frac{15731}{53725}, \quad \hat{p}_{44} := \frac{37994}{53725},$
M	$\hat{p}_{45} := 1,$
X	$\hat{p}_{46} := \frac{6196}{87035}, \quad \hat{p}_{47} := \frac{80839}{87035},$
Y	$\hat{p}_{48} := 1,$
Z	$\hat{p}_{49} := \frac{2812}{55123}, \quad \hat{p}_{50} := \frac{52311}{55123},$
N	$\hat{p}_{51} := \frac{7737}{17437}, \quad \hat{p}_{52} := \frac{9700}{17437},$
U	$\hat{p}_{53} := \frac{109939}{518817}, \quad \hat{p}_{54} := \frac{408878}{518817},$

Table 2 Floating point approximations of the probabilities (relative frequencies) for the production rules of the SCFG $\hat{\mathcal{G}}_{sto}$ (rounded to three decimal places)

Nonterminal *Nt*	Probabilities of Rules with Premise *Nt*
S′	$\hat{p}_1 := 1.000,$
E	$\hat{p}_2 := 0.021, \quad \hat{p}_3 := 0.979,$
S	$\hat{p}_4 := 0.014, \quad \hat{p}_5 := 0.986,$
T	$\hat{p}_6 := 0.868, \quad \hat{p}_7 := 0.132,$
C	$\hat{p}_8 := 0.096, \quad \hat{p}_9 := 0.904,$
A	$\hat{p}_{10} := 1.000$
L	$\hat{p}_{11} := 0.763, \quad \hat{p}_{12} := 0.040, \quad \hat{p}_{13} := 0.019, \quad \hat{p}_{14} := 0.037,$
	$\hat{p}_{15} := 0.007, \quad \hat{p}_{16} := 0.086, \quad \hat{p}_{17} := 0.048,$
G	$\hat{p}_{18} := 0.304, \quad \hat{p}_{19} := 0.188, \quad \hat{p}_{20} := 0.308, \quad \hat{p}_{21} := 0.200,$
D	$\hat{p}_{22} := 1.000$
B	$\hat{p}_{23} := 0.390, \quad \hat{p}_{24} := 0.610,$
F	$\hat{p}_{25} := 0.057, \quad \hat{p}_{26} := 0.341, \quad \hat{p}_{27} := 0.602,$
H	$\hat{p}_{28} := 0.201, \quad \hat{p}_{29} := 0.799,$
P	$\hat{p}_{30} := 0.109, \quad \hat{p}_{31} := 0.214, \quad \hat{p}_{32} := 0.201, \quad \hat{p}_{33} := 0.476,$
Q	$\hat{p}_{34} := 0.171, \quad \hat{p}_{35} := 0.829,$
R	$\hat{p}_{36} := 0.415, \quad \hat{p}_{37} := 0.585,$
V	$\hat{p}_{38} := 1.000$
W	$\hat{p}_{39} := 1.000$
O	$\hat{p}_{40} := 1.000$
J	$\hat{p}_{41} := 0.324, \quad \hat{p}_{42} := 0.676,$
K	$\hat{p}_{43} := 0.293, \quad \hat{p}_{44} := 0.707,$
M	$\hat{p}_{45} := 1.0000$
X	$\hat{p}_{46} := 0.071, \quad \hat{p}_{47} := 0.929,$
Y	$\hat{p}_{48} := 1.0000$
Z	$\hat{p}_{49} := 0.051, \quad \hat{p}_{50} := 0.949,$
N	$\hat{p}_{51} := 0.444, \quad \hat{p}_{52} := 0.556,$
U	$\hat{p}_{53} := 0.212, \quad \hat{p}_{54} := 0.788.$

90% (resp. 50%) portion of the original training set and re-estimated the probabilities of all the grammar rules. This process was iterated 40 times, resulting in a sample of 40 parameter sets. Finally, for each parameter we determined its variance along this sample of size 40. The corresponding values lay between 0 (resulting for intermediate symbols without alternatives; for whose productions a probability of 1 is predetermined) and 2.87652×10^{-6} (resp. 2.86242×10^{-5}). We can conclude that over-fitting is no issue in connection with our sophisticated grammar and the training set used.

Derivation of the Algorithm

The elaborate SCFG $\hat{\mathcal{G}}_{sto}$ is appropriate for being used as the basis for the desired weighed unranking method: after having determined the RNF of this SCFG and the corresponding weighted combinatorial classes, we easily find a recursion for the `size` function (in the same ways as discussed in Example `App-`.4). Then, we can use the resulting weighted class sizes for the straightforward construction of the desired unranking algorithm.

In fact, for the construction of the complete algorithm, we simply have to use Algorithms 1 to 4 (Unranking of neutral classes, atomic classes, disjoint unions and cartesian products, respectively) and Algorithm 6 (Unranking of weighted classes) given in [20] as subroutines. However, to improve the worst-case complexity of the resulting unranking procedure from $\mathcal{O}(n^2)$ to $\mathcal{O}\left(n \cdot \log(n)\right)$ by using the *boustrophedonic order* instead of the *sequential order*, a simple change in Algorithm 4 (Unranking of cartesian products) is neccessary (see e.g. [7]).

A random RNA secondary structure of size n can easily be computed by drawing a random number $i \in \{0, \ldots, \text{size}(\mathcal{L}, n) - 1\}$ and then unranking the ith structure of size n. The worst-case runtime complexity of this procedure is equal to that of unranking and is thus given by $\mathcal{O}\left(n \cdot \log(n)\right)$ when using the boustrophedonic order. By repeating this procedure m times, a set of m (not necessarily distinct) random RNA secondary structures of size n can be generated in time $\mathcal{O}\left(m \cdot n \cdot \log(n)\right)$, where a preprocessing time of $\mathcal{O}(n^2)$ is required for the computation of all (weighted) class sizes up to input length n.

A complete and detailed description of the derivation of our weighted unranking algorithm for (SSU and LSU r)RNA secondary structures can be found in the Appendix, since it is too comprehensive to be presented here and the different steps for its generation correspond to those described in [20].

Availability of Software

It may be of interest to the reader that this non-uniform random generation algorithm for RNA secondary structures has been implemented as a webservice which is accessible to the scientific community under http://wwwagak.cs.uni-kl.de/NonUniRandGen. Since it is relevant for researchers to have methods available for generating random structures that are realistic for a particular investigation, this webservice is also capable of allowing the user to specify the distribution from which the corresponding structures should be sampled (in the form of a set of secondary structure samples from which the parameters for our grammars are inferred). Furthermore, our Mathematica source code used to implement the webservice can be downloaded from our website and used under GNU public licence.

Discussion

The purpose of this section is to analyze the quality of randomly generated structures by considering some experimental results.

Parameters for Structural Motifs

As a first step, we decided to consider several important parameters related to particular structural motifs of RNA secondary structure and compare the observed statistical values derived from a native sample (here our biological database, i.e. the set of real-life RNA data that we used for deriving the distribution and thus the weights for the unranking algorithm) to those derived from a corresponding random sample (i.e. a set of random structures generated by our algorithm). In order to obtain an appropriate random sample, we have generated exactly one random structure of size n for each native RNA structure of size n given in our database, such that for each occurring size n, the random sample and the native sample contain the same number of structures having this size.

The determined results are presented in Table 3. Comparing the specific values of all different parameters, we can guess that our algorithm produces random RNA secondary structures that are, related to the different structural motifs and thus related to the

Table 3 Expectation and variance of important parameters related to particular structural motifs of RNA secondary structure

Parameter	Expected Value		Variance	
	Random	Native	Random	Native
num_{unp}	848.179	839.956	98964.7	103426.
num_{bps}	420.848	424.96	27785.3	31310.9
num_{urs}	179.73	181.822	4959.96	5117.47
num_{e}	1.	1.	0.	0.
num_{h}	36.6983	36.4818	196.935	185.596
num_{s}	321.18	324.26	16538.8	19343.4
num_{b}	20.6061	20.5782	87.1894	50.3103
num_{i}	26.1442	26.538	125.66	194.769
num_{m}	16.2197	17.1018	57.8874	41.0261
num_{hel}	99.6683	100.7	1549.24	1492.84
unp_{e}	106.014	79.8382	4039.69	3897.61
unp_{h}	6.93534	6.93188	18.4264	77.464
unp_{s}	–	–	–	–
unp_{b}	1.9948	1.99596	3.10283	6.87868
unp_{i}	7.14617	7.08869	16.5725	31.1197
unp_{m}	16.0122	16.2577	87.4906	195.497
unp_{hel}	–	–	–	–
bps_{e}	9.41479	6.94105	29.1956	6.30949
bps_{h}	–	–	–	–
bps_{s}	1.	1.	0.	0.
bps_{b}	1.	1.	0.	0.
bps_{i}	1.	1.	0.	0.
bps_{m}	2.68212	2.72734	1.12921	1.21643
bps_{hel}	4.22249	4.22006	13.6266	5.52299

Values are derived from a native sample (our biological database) and from a random sample, respectively. num_xdenotes the number of occurrences of motif $_x$in one secondary structure and unp_x(bps_x) denotes the number of accessible unpaired bases (base pairs) in one substructure of type *x*. unp, bps, urs denote unpaired bases, base pairs and unpaired regions, whereas *e*, *h*, *s*, *b*, *i*, *m*, *hel* denote exterior loop, hairpin loop, stacked pair, bulge loop, interior loop, multiloop and helix, respectively.

expected shape of such structures, in most cases realistic. Obviously, this is a major improvement over existing approaches for the random generation of secondary structures of a given input size n (where the corresponding specific RNA sequence is *not* known, but only its length n), as those (sequence-independent) methods are only capable of generating structures uniformly at random for input size n. Furthermore, with the SCFG model used here, we have an new model for RNA secondary structures at hand which realistically reflects the structure of an RNA molecule and its basic structural motifs.

Related Free Energies

For further investigation on the accuracy of our random generator, we take on a completely different point of view and consider thermodynamics. The reason behind this idea is that if an RNA secondary structure model induced by a SCFG shows a realistic behaviour (expectation and variance) with respect to minimum free energy, then it is rather likely that our grammar also shows a realistic picture for all the different structural motifs of a molecule's folding (as the free energy of a molecule's structure is defined as the sum of the energy contributions of all its substructures).

Since we do not know the corresponding RNA sequences for the randomly generated structures, we can not use one of the common sequence-dependent thermodynamic models for RNAs. Therefore, we decided to consider both the *static* and *dynamic* free energy models (in the static model, averaged free energy contributions for the distinguished structural motifs are considered which can easily be derived from the training data (by sequence counting). These averaged values actually represent the free energy contributions that have to be added for the respective whole substructures. For the dynamic model, corresponding average values for length-dependent free energy contributions (that depend on the number of unpaired or paired bases within particular substructures) are added for each component (unpaired base or base pair) in the respective motifs, such that in contrast to the static model, substructures of different lengths are assigned different free energy values) defined in [21] for RNA secondary structures with unknown sequence. These models are based on the well-known Turner energy model [22,23] and model parameters have been derived from the same biological database (of SSU and LSU rRNAs) that we consider in this article. In fact, both models have turned out to show a realistic behaviour and can therefore be used to judge the quality of random structures generated by our algorithm.

Unquantified Results

Similar to [21], we denote the free energy of a given secondary structure $s \in \mathcal{L}$ according to the static and dynamic model by $g_{stat}(s)$ and $g_{dyn}(s)$, respectively. Moreover, the expected free energy and corresponding variance that have been analytically derived in that paper for any $n > 0$ are denoted by $\mu_{energy,n} := \mathbb{E}\left[energy(s) \mid \text{size}(s) = n\right]$ and $\sigma^2_{energy,n} := \mathbb{V}\left[energy(s) \mid \text{size}(s) = n\right]$, respectively, where $energy \in \{g_{stat}, g_{dyn}\}$. The corresponding confidence interval for $n > 0$ and $k > 1$, which contains at least $\left(100 - \frac{100}{k^2}\right)$ percent of the energies in $\{energy(s) \mid s \in \mathcal{L}^n\}$ is denoted by $I_{energy,n}(k) := (\mu_{energy,n} - k\sigma_{energy,n}, \mu_{energy,n} + k\sigma_{energy,n})$. As these analytical energy results from [21] and our unranking algorithm have been derived from the same database of real-life RNA data and by modeling the same class $\mathcal{L}$ of structures via very similar SCFGs, it seems adequate to use them for comparisons with the energies of our randomly generated structures.

Before we start with our comparisons, note that for any sample set $\mathcal{S}$ of secondary structures, we can calculate the corresponding energy points $EP(\mathcal{S}, energy) := \{(\text{size}(s), energy(s)) \mid s \in \mathcal{S}\}$, where $energy \in \{g_{stat}, g_{dyn}\}$. Obviously, we can also compute the corresponding "average energy points" $AvEP(\mathcal{S}, energy) := \left\{(n, \mu_n := \frac{1}{\text{card}(\mathcal{S}^n)} \sum_{s \in \mathcal{S}^n} energy(s)) \mid \mathcal{S}^n \neq \emptyset\right\}$ and the corresponding "energy variance points" $VarEP(\mathcal{S}, energy) := \left\{(n, \sigma^2_n := \frac{1}{\text{card}(\mathcal{S}^n)} \sum_{s \in \mathcal{S}^n} \left(\mu_n - energy(s)\right)^2 \mid \mathcal{S}^n \neq \emptyset\right\}$, respectively. In the sequel, we will denote a random sample generated by our algorithm by $\mathscr{R}$ and a native sample (biological database) by $\mathcal{N}$.

In order to obtain an appropriate random sample for our energy comparisons, we derived a large set of random structures by generating 1000 RNA secondary structures for each of the sizes $n \in \{500,1000,1500,..., 5000, 5500\}$ with our weighted unranking algorithm. To compare the energies of our randomly generated structures to the corresponding confidence interval(s), we decided to consider any $k \in \{\sqrt{2}, 2, \sqrt{10}, \sqrt{20}\}$, meaning the probability that the free energy of a random RNA secondary structure of size n lies within the corresponding interval is greater than 0.5, 0.75, 0.9, and 0.95, respectively.

Figure 4 shows a plot of the corresponding four confidence intervals (analytically derived, related to our biological data) along with the energy points for our random sample and for our native database, respectively, under the assumption of the static energy model. The corresponding plots for the dynamic energy model are shown in Figure 5. Looking at both figures, we immediately see that the energies for our set of randomly generated RNA secondary structures seem to fit to the ones for the considered RNA database and also to the corresponding analytically obtained energy results from [21]. This observation becomes even more clear by considering Figures 6 and 7. There, we compare the previously introduced “average energy points” and “energy variance points” to the analytically determined expected free energy and corresponding variance from [21], respectively.

Quantified Results

The previously considered energy comparisons have been presented only by unquantified plots. This may not be very satisfying, since it is obvious that the free energy would decrease with structure size and aside from this, it could have been expected that for large randomly generated sets of structures of a given size, the average energy and corresponding variance fit the analytically obtained energy results derived under the assumption of a basically equivalent SCFG model for secondary structures. Therefore, there is a need to consider some sort of quantification and additionally present corresponding quantified comparison results. What really matters is the degree to which the energy ranges of the random structures agree, in distribution, with our biological database. This means we have to find out if the energies related to a random sample (generated by our unranking method) and those related to a native sample (given by the structures in our biological database) come from a common distribution.

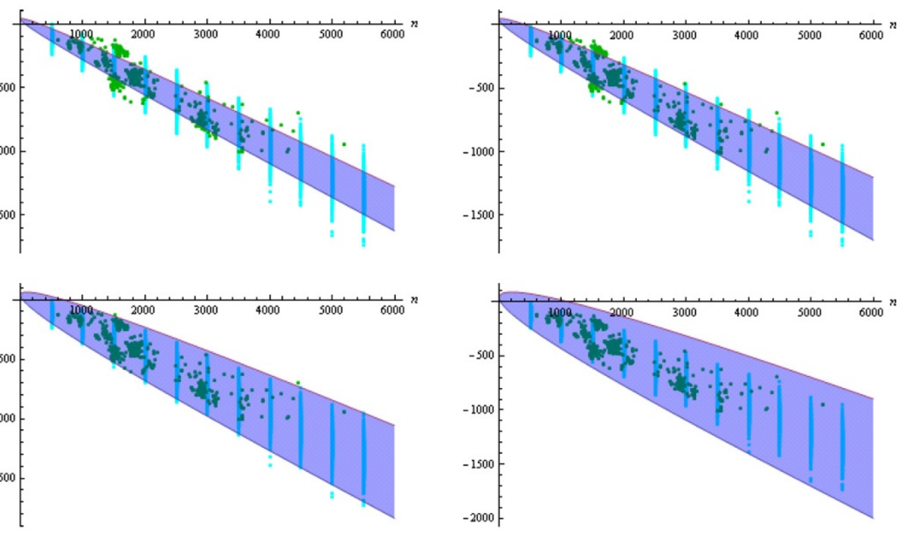

Figure 4 Plots of the confidence intervals $I_{g_{stat},n(k)}$. Intervals are shown for the static energy model (blue), for $k \in \{\sqrt{2}, 2, \sqrt{10}, \sqrt{20}\}$ (top left to bottom right), together with the corresponding energy points $EP(\mathcal{R}, g_{stat})$for the random sample (cyan) and $EP(\mathcal{N}, g_{stat})$ for the native sample (green).

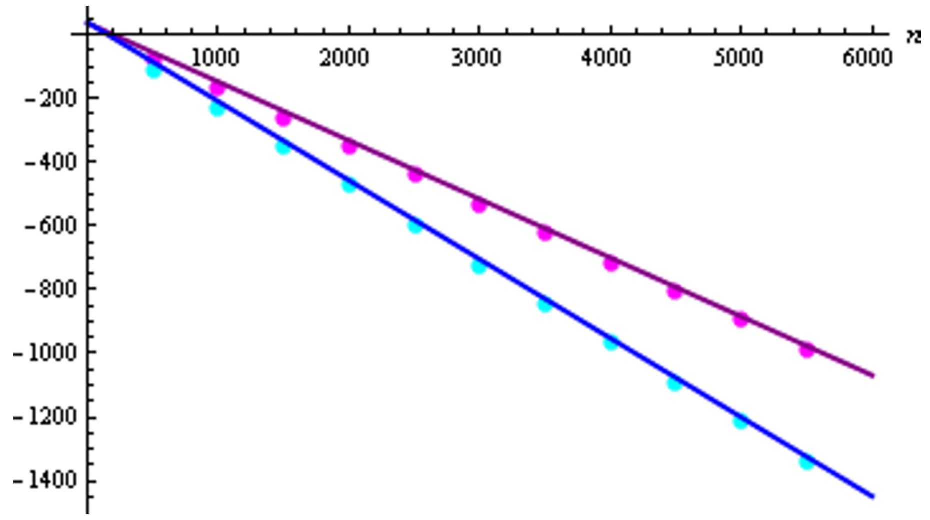

Figure 6 Plot of expectations of the free energy. Plots show $\mu_{g_{stat},n}$ (blue) and $\mu_{g_{dyn},n}$ (purple) of a random RNA secondary structure of size *n*, together with the "average energy points" $AvEP(\mathcal{R}, g_{stat})$(cyan) and $AvEP(\mathcal{R}, g_{dyn})$ (magenta) for the random sample.

Consequently, we have to consider the energies of a random sample and those of a native one as two independent sets of values and determine the extend to which their distributions coincide, or in other words to test for significant differences between these two sets. For this reason, we decided to apply one of the most common (non-parametric) significance tests known from statistics, the so-called *Mann-Whitney U-test* [48], which is widely used as statistical hypothesis test for assessing whether two independent samples of observations (with arbitrary sample sizes) come from the same distribution. It is also known as the *Wilcoxon rank-sum test* [49] which however can only be applied for equal sample sizes.

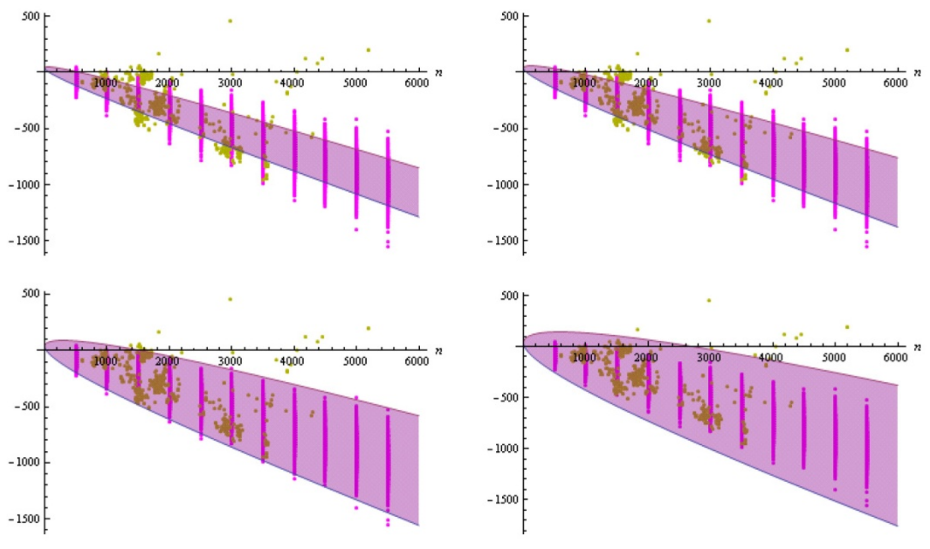

Figure 5 Plots of the confidence intervals $I_{g_{dyn},n}(k)$. Intervals are shown for the dynamic energy model (purple), for $k \in \{\sqrt{2}, 2, \sqrt{10}, \sqrt{20}\}$ (top left to bottom right), together with the corresponding energy points $EP(\mathcal{R}, g_{dyn})$for the random sample (magenta) and $EP(\mathcal{N}, g_{dyn})$ for the native sample (yellow).

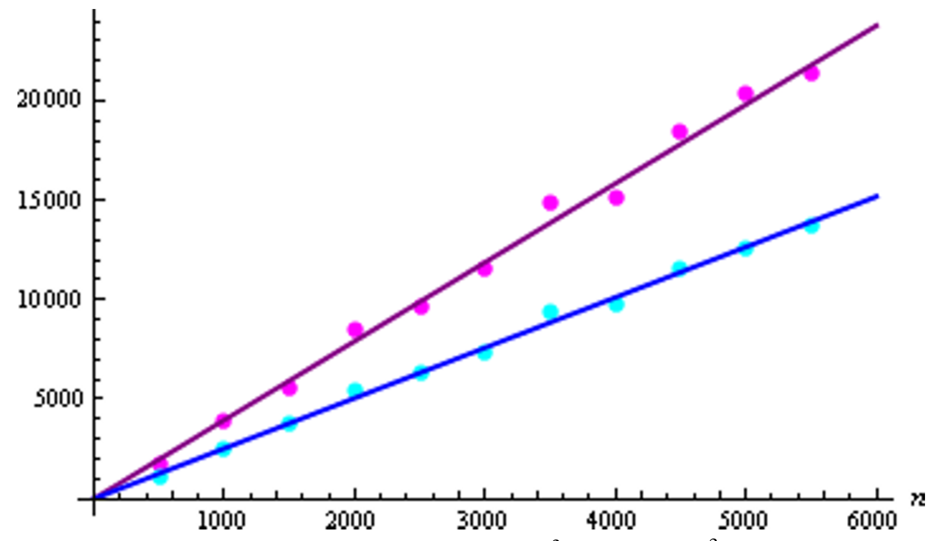

Figure 7 Plot of variances of the free energy. Plots show $\sigma^2_{g_{stat},n}$ (blue) and $\sigma^2_{g_{dyn},n}$ (purple) of a random RNA secondary structure of size *n*, together with the "energy variance points" $VarEP(\mathcal{R}, g_{stat})$ (cyan) and $VarEP(\mathcal{R}, g_{dyn})$ (magenta) for the random sample.

Formally, this test is used to check whether the *null hypothesis* N_0 - which states that the two independent samples X and Y are identically distributed (i.e. $F(X) = F(Y)$) - can be accepted or else, has to be rejected. More specifically, the result of such a test, the so-called *p-value*, is a probability answering the following question: If the two samples really have the same distribution, what is the probability that the observed difference is due to chance alone? In other words, were the deviations (differences between the two samples) the result of chance, or were they due to other factors and how much deviation can occur before one must conclude that something other than chance causes the differences? The *p*-value is called *statistically significant* if it is unlikely that the differences occurred by chance alone, according to a preliminary chosen threshold probability, the *significance level* α (common choices are e.g. $\alpha \in \{0.10, 0.05, 0.01\}$). If $p \geq \alpha$, the deviation is small enough that chance alone accounts for it; this is within the range of acceptable deviation. If $p < \alpha$, we must conclude that some factor other than chance causes the deviation to be so great, this will lead us to decide that the two sets come from different distributions.

For our analysis, we again decided to generate the same numbers of random structures for any size as are given for this size in our biological database, such that random and native sample contain the same numbers of structures for any occurring size (and hence the sample sizes are equal). Moreover, note that the unquantified results presented in Figures 4 and 5 might yield the assumption that for any structure size, some energy values of randomly generated structures are scattered too widely around the corresponding expected value, such that those randomly drawn secondary structures can not be considered realistic (neither with respect to thermodynamics nor with respect to structural composition and expected shape). In an attempt to disprove that assumption, we decided to perform a series of Wilcoxon tests by considering a number of different random samples. These samples are created by obeying a specified energy-based rejection scheme: Do not add a randomly generated structure of a given size to the sample if its free energy (according to the static or dynamic model or according to both models) lies outside the corresponding confidence interval(s). Formally, for any preliminary chosen value $k > 1$, a generated structure $s \in \mathcal{L}^n$ is added to the random sample iff

$$[g_{stat} \in I_{g_{stat},n}(k) \text{ (variant "static")}] \text{ or } [g_{dyn} \in I_{g_{dyn},n}(k) \text{ (variant "dynamic")}] \text{ or }$$
$$[g_{stat} \in I_{g_{stat},n}(k) \text{ and } g_{dyn} \in I_{g_{dyn},n}(k) \text{ (variant "both")}];$$

otherwise it is rejected. This means we accept only a specified deviation of the energy *energy* (s) of the random structure s from the corresponding expected free energy $\mu_{energy,n}$ and reject structures whose energy differs too much from the expected value. Note that for $k = \infty$ (confidence interval $I_{energy,n}(k)$ contains 100 percent of the energies *energy*(s) of all $s \in \mathcal{L}^n$), no structures are rejected. Hence, in this case, the corresponding random sample corresponds to the usual (unrestricted) output of our algorithm.

The Wilcoxon test results for our native sample together with any of a number of random sample sets generated in the previously described restricted manner, respectively, can be found in Table 4. As we can see, the best results are achieved for the unrestricted sample sets, where all free energies of randomly generated structures were allowed during the sample creation process. Moreover, these two results (for the unrestricted case $k = \infty$) are not statistically significant when considering the common significance level $\alpha = 0.05$, that is in both cases, we can assume that the energies of the random structures and those of the biological data follow a common distribution. These observations indicate that our weighted unranking algorithm produces random RNA secondary structures that are - related to the free energy of such structures (in expectation and variation) - in expectation realistic.

Besides that, it is obvious that the computed p-values are much better for the dynamic energy model than for the static one. This underlines the suggestion made in [21] that, although both energy models have been proven to be realistic, due to the more realistic variation of free energies connected to varying loop length, the dynamic model should be used for possible applications. Since at least for the dynamic model, the random data fit very nicely with the native data, we can conclude that structures generated by our non-uniform random generation algorithm behave realistic with respect to free energies and - as the energy of the overall structure is assumed to be equal to the sum of the substructure energies - rather likely also with respect to appearance of the different structural motifs of RNA molecules.

Conclusion

Altogether, we can finally conclude that the non-uniform random generation method proposed in this article produces appropriate output and may thus be used (for research issues as well as for practical applications) to generate random RNA secondary structures. In fact, for any arbitrary type of (pseudoknot-free) RNA, a corresponding random sampler can be derived in the presented way. Actually, our webservice can be used for generating random secondary structures of any specified type of RNA. It just requires a database of known structures for the respective RNA type as input.

Note that in this work, we abstract from sequence and consider only the structure size as input for our algorithm. Thus, an interesting problem for future research would

Table 4 Significance results for statistical hypothesis testing, computed by the Wilcoxon rank-sum method

Chosen Value of *k*	Percent Within Corr. Interval	Models Used for Rejection	Model for Native Energies	Model for Random Energies	Resulting Wilcoxon *p*-Value (approx.)
$\frac{10}{3\sqrt{11}} \approx 1.00504$	1	Dynamic	Dynamic	Dynamic	0.0008438
		Static	Static	Static	$1.872 \cdot 10^{-9}$
		Both	Dynamic	Dynamic	0.000507
		Both	Static	Static	$1.851 \cdot 10^{-10}$
$\frac{2\sqrt{5}}{\sqrt{19}} \approx 1.02598$	5	Dynamic	Dynamic	Dynamic	0.001567
		Static	Static	Static	$1.454 \cdot 10^{-10}$
		Both	Dynamic	Dynamic	0.0002654
		Both	Static	Static	$1.009 \cdot 10^{-9}$
$\frac{\sqrt{10}}{3} \approx 1.05409$	10	Dynamic	Dynamic	Dynamic	0.001374
		Static	Static	Static	$3.526 \cdot 10^{-9}$
		Both	Dynamic	Dynamic	0.0004116
		Both	Static	Static	$9.018 \cdot 10^{-10}$
$\frac{2}{\sqrt{3}} \approx 1.15470$	25	Dynamic	Dynamic	Dynamic	0.003618
		Static	Static	Static	$2.530 \cdot 10^{-7}$
		Both	Dynamic	Dynamic	0.001228
		Both	Static	Static	$1.162 \cdot 10^{-7}$
$\sqrt{2} \approx 1.41421$	50	Dynamic	Dynamic	Dynamic	0.02394
		Static	Static	Static	$1.278 \cdot 10^{-6}$
		Both	Dynamic	Dynamic	0.001389
		Both	Static	Static	$1.515\ 10^{-7}$
2	75	Dynamic	Dynamic	Dynamic	0.1184
		Static	Static	Static	0.001034
		Both	Dynamic	Dynamic	0.0495
		Both	Static	Static	0.0009445
∞	100	–	Dynamic	Dynamic	0.4007
		–	Static	Static	0.08961

be to find a way to extend the presented realistic SCFG model to additionally deal with RNA sequence. In fact, this work and especially the considered elaborate SCFG could mark some sort of stepping stone towards new stochastic RNA secondary structure prediction methods realized by statistical random sampling.

Appendix

How to Construct a Weighted Unranking Algorithm from a Given SCFG

The purpose of this section is to give a rather small example for applying the construction scheme described in detail in [20] to proceed from an arbitrary SCFG to reweighted normal form (RNF) and then to the corresponding weighted combinatorial classes which allow for non-uniform generation by means of unranking.

Example App-.4. Let us consider the SCFG $\mathcal{G}_d$, which contains the following rules:

$$\begin{aligned}
&w_1 : S \to B,\\
&w_2 : B \to \mathbf{(}B\mathbf{)},\ w_3 : B \to \mathbf{|}C,\\
&w_4 : C \to \epsilon, \quad w_5 : C \to \mathbf{|}C.
\end{aligned}$$

To apply the approach presented in [20] to transform a given SCFG to RNF, the grammar needs to be ϵ-free and loop-free. Thus, we first have to transform grammar $\mathcal{G}_d$ into the following one:

$$\begin{aligned}&\widehat{w}_1{:}S \to B,\\ &\widehat{w}_2{:}B \to \mathbf{(}B\mathbf{)},\ \widehat{w}_3{:}B \to C,\\ &\widehat{w}_4{:}C \to \mathbf{|},\quad \widehat{w}_5{:}C \to \mathbf{|}C.\end{aligned}$$

The transformation of $\mathcal{G}_d$ into RNF now works as follows: First, we have to gather all possible chains $A \to A_1 \to A_2 \to \ldots \to \alpha$, where $A \neq S$ and $|\alpha| = 1$. These chains are $B \to C$, $B \to C \to \mathbf{|}$ and $C \to \mathbf{|}$; the rules $B \to C$ and $C \to \mathbf{|}$ are then removed. Second, we have to replace each of these chains by a specific new rule. In fact, we have to add $B^{C,\epsilon} \to C$, $B^{\mathbf{|},C} \to \mathbf{|}$ and $C^{\mathbf{|},\epsilon} \to \mathbf{|}$ to the new set of productions. Consequently, our new rule set is now given by

$$\begin{aligned}&\widehat{w}_1{:}S \to B,\\ &\widehat{w}_2{:}B \to \mathbf{(}B\mathbf{)},\\ &\widehat{w}_5{:}C \to \mathbf{|}C,\\ &\ 1{:}B^{C,\epsilon} \to C,\ 1{:}B^{\mathbf{|},C} \to \mathbf{|},\ 1{:}C^{\mathbf{|},\epsilon} \to \mathbf{|}.\end{aligned}$$

Third, for each occurrence of a non-terminal symbol A in the conclusion of a production and each previously added new rule $A^{\alpha,A_1A_2\cdots} \to \alpha$ corresponding to a chain $A \to A_1 \to A_2 \to \ldots \to \alpha$, add a specific new rule. This way, we obtain the following production set:

$$\begin{aligned}&\widehat{w}_1{:}S \to B,\quad \widehat{w}_1 \cdot \widehat{w}_3{:}S \to B^{C,\epsilon},\quad \widehat{w}_1 \cdot \widehat{w}_3 \cdot \widehat{w}_4{:}S \to B^{\mathbf{|},C},\\ &\widehat{w}_2{:}B \to \mathbf{(}B\mathbf{)},\ \widehat{w}_2 \cdot \widehat{w}_3{:}B \to \mathbf{(}B^{C,\epsilon}\mathbf{)},\ \widehat{w}_2 \cdot \widehat{w}_3 \cdot \widehat{w}_4{:}B \to \mathbf{(}B^{\mathbf{|},C}\mathbf{)},\\ &\widehat{w}_5{:}C \to \mathbf{|}C,\ \widehat{w}_5 \cdot \widehat{w}_4{:}C \to \mathbf{|}C^{\mathbf{|},\epsilon},\\ &\ 1{:}B^{C,\epsilon} \to C,\qquad 1{:}B^{\mathbf{|},C} \to \mathbf{|},\qquad\qquad 1{:}C^{\mathbf{|},\epsilon} \to \mathbf{|}.\end{aligned}$$

Fourth, each intermediate symbol that no longer occurs as premise in any of the productions has to be removed and fifth, each production of the form $S \to \alpha$, where S is the axiom and $|\alpha| > 1$ has to be changed in a specific way. However, since in our case, there is obviously nothing left to do, the transformation of $\mathcal{G}_d$ into RNF is finished.

For $\mathcal{G}_d$ (in RNF), where all production weights are rational, we can determine the common denominator s of the weights of productions with premise S, as well as the common denominator c of the weights of the remaining productions (i.e., of the productions with premise B or C). Then, the reweighting of the production rules of (the RNF of) $\mathcal{G}_d$ is done by multiplying the weights of productions with source S by s, and the weights of the other productions $A \to \alpha$, where $A \neq S$, by the factor $c^{|\alpha|-1}$. After that, we obtain the following reweighted grammar $\mathcal{G}'_d$:

$$\begin{aligned}&w'_1{:}S \to B,\quad w'_2{:}S \to B^{C,\epsilon},\quad w'_3{:}S \to B^{\mathbf{|},C},\\ &w'_4{:}B \to \mathbf{(}B\mathbf{)},\ w'_5{:}B \to \mathbf{(}B^{C,\epsilon}\mathbf{)},\ w'_6{:}B \to \mathbf{(}B^{\mathbf{|},C}\mathbf{)},\\ &w'_7{:}C \to \mathbf{|}C,\ \ w'_8{:}C \to \mathbf{|}C^{\mathbf{|},\epsilon},\\ &\ 1{:}B^{C,\epsilon} \to C,\ \ 1{:}B^{\mathbf{|},C} \to \mathbf{|},\qquad 1{:}C^{\mathbf{|},\epsilon} \to \mathbf{|},\end{aligned}$$

where each W'_i, $1 \leq i \leq 8$, is integral.

The (now weighted) grammar can easily be translated into a corresponding admissible specification, which includes the weighting of all involved combinatorial (sub) classes, as described earlier. For the reweighted grammar $\mathcal{G}'_d$, this specification is given by the following equations:

$$\begin{array}{lll}
\mathcal{S}_1 = \mathcal{B}, & \mathcal{S}_2 = \mathcal{B}^{C,\varepsilon}, & \mathcal{S}_3 = \mathcal{B}^{|,C}, \\
\mathcal{B}_1 = \mathcal{Z}_{(\times\mathcal{B}\times\mathcal{Z})}, & \mathcal{B}_2 = \mathcal{Z}_{(\times\mathcal{B}^{C,\varepsilon}\times\mathcal{Z})}, & \mathcal{B}_3 = \mathcal{Z}_{(\times\mathcal{B}^{|,C}\times\mathcal{Z})}, \\
\mathcal{C}_1 = \mathcal{Z}_{|} \times \mathcal{C}, & \mathcal{C}_2 = \mathcal{Z}_{|} \times \mathcal{C}^{|,\varepsilon}, & \\
\mathcal{B}^{C,\varepsilon} = \mathcal{C}, & \mathcal{B}^{|,C} = \mathcal{Z}_{|}, & \mathcal{C}^{|,\varepsilon} = \mathcal{Z}_{|},
\end{array}$$

$$\begin{aligned}
\mathcal{S} &= w'_1 \cdot \mathcal{S}_1 + w'_2 \cdot \mathcal{S}_2 + w'_3 \cdot \mathcal{S}_3, \\
\mathcal{B} &= w'_4 \cdot \mathcal{B}_1 + w'_5 \cdot \mathcal{B}_2 + w'_6 \cdot \mathcal{B}_3, \\
\mathcal{C} &= w'_7 \cdot \mathcal{C}_1 + w'_8 \cdot \mathcal{C}_2,
\end{aligned}$$

which can be simplified in the following way:

$$\begin{array}{lll}
\mathcal{B}_1 = \mathcal{Z}_{(} \times \mathcal{B} \times \mathcal{Z}_{)}, & \mathcal{B}_2 = \mathcal{Z}_{(} \times \mathcal{C} \times \mathcal{Z}_{)}, & \mathcal{B}_3 = \mathcal{Z}_{(} \times \mathcal{Z}_{|} \times \mathcal{Z}_{)}, \\
\mathcal{C}_1 = \mathcal{Z}_{\_} \times \mathcal{C}, & \mathcal{C}_2 = \mathcal{Z}_{\_} \times \mathcal{Z}_{\_}, &
\end{array}$$

$$\begin{aligned}
\mathcal{S} &= w'_1 \cdot \mathcal{B} + w'_2 \cdot \mathcal{C} + w'_3 \cdot \mathcal{Z}_{|}, \\
\mathcal{B} &= w'_4 \cdot \mathcal{B}_1 + w'_5 \cdot \mathcal{B}_2 + w'_6 \cdot \mathcal{B}_3, \\
\mathcal{C} &= w'_7 \cdot \mathcal{C}_1 + w'_8 \cdot \mathcal{C}_2.
\end{aligned}$$

As described earlier, this specification (with weighted classes) derived from reweighted grammar $\mathcal{G}'_d$ transforms immediately into a recursion for the function size of all needed combinatorial classes. For $\mathcal{G}'_d$, the recursion for the function size has the following form:

$$\text{size}(\mathcal{I}, n) := \begin{cases}
\text{size}(\mathcal{B}, n-2) & \mathcal{I} = \mathcal{B}_1, \\
\text{size}(\mathcal{C}, n-2) & \mathcal{I} = \mathcal{B}_2, \\
1 & \mathcal{I} = \mathcal{B}_3 \text{ and } n = 3, \\
\text{size}(\mathcal{C}, n-1) & \mathcal{I} = \mathcal{C}_1, \\
1 & \mathcal{I} = \mathcal{C}_2 \text{ and } n = 2, \\
w'_1 \cdot \text{size}(\mathcal{B}, n) + w'_2 \cdot \text{size}(\mathcal{C}, n) + w'_3 \cdot 1 & \mathcal{I} = \mathcal{S} \text{ and } n = 1, \\
w'_1 \cdot \text{size}(\mathcal{B}, n) + w'_2 \cdot \text{size}(\mathcal{C}, n) + w'_3 \cdot 0 & \mathcal{I} = \mathcal{S} \text{ and } n \neq 1, \\
w'_4 \cdot \text{size}(\mathcal{B}_1, n) + w'_5 \cdot \text{size}(\mathcal{B}_2, n) + w'_6 \cdot \text{size}(\mathcal{B}_3, n) & \mathcal{I} = \mathcal{B}, \\
w'_7 \cdot \text{size}(\mathcal{C}_1, n) + w'_8 \cdot \text{size}(\mathcal{C}_2, n) & \mathcal{I} = \mathcal{C}, \\
0 & \text{else.}
\end{cases}$$

This recursive size function (with weighted class sizes) can now be used for the straightforward construction of a corresponding algorithm for the non-uniform generation of elements of $\mathcal{L}(\mathcal{G}_d)$ by means of unranking, as proposed in [20].

Derivation of the Algorithm

In this section, we give a complete and detailled description of the derivation of our weighted unranking algorithm for RNA secondary structures. The different steps are made according to the approach described in [20] to get an unranking algorithm that generates random RNA secondary structures of a given size n according to the distribution on all these structures.

Considered (unambiguous, ϵ-free and loop-free) SCFG

First, note that in [21], to obtain the stochastic model for RNA secondary structures derived from real-world RNA data, the following unambiguous SCFG which unambiguously generates exactly the language $\mathcal{L}$ given in Definition 0.9 has been used:

Definition App-.11. The unambiguous SCFG $\mathcal{G}_{\text{sto}}$ generating exactly the language $\mathcal{L}$ is given by $\mathcal{G}_{\text{sto}} = (I_{\mathcal{G}_{\text{sto}}}, \sum_{\mathcal{G}_{\text{sto}}}, R_{\mathcal{G}_{\text{sto}}}, S)$, where

$$I_{\mathcal{G}_{sto}} = \{S, T, C, A, L, G, B, F, H, P, Q, R, J, K, M, N, U\},$$

$\Sigma_{\mathcal{G}_{\text{sto}}} = \{(,), |\}$ and $\mathcal{R}_{\mathcal{G}_{\text{sto}}}$ contains exactly the following rules:

$$
\begin{array}{llll}
p_1{:}S \to TAC, & & & \\
p_2{:}T \to TAC, & p_3{:}T \to C, & & \\
p_4{:}C \to C|, & p_5{:}C \to \epsilon, & & \\
p_6{:}A \to (L), & & & \\
p_7{:}L \to (L), & p_8{:}L \to M, & p_9{:}L \to P, & p_{10}{:}L \to Q, \\
p_{11}{:}L \to R, & p_{12}{:}L \to F, & p_{13}{:}L \to G, & \\
p_{14}{:}G \to (L)|, & p_{15}{:}G \to (L)B||, & p_{16}{:}G \to |(L), & p_{17}{:}G \to ||B(L), \\
p_{18}{:}B \to B|, & p_{19}{:}B \to \epsilon, & & \\
p_{20}{:}F \to |||, & p_{21}{:}F \to ||||, & p_{22}{:}F \to |||||H, & \\
p_{23}{:}H \to H|, & p_{24}{:}H \to \epsilon, & & \\
p_{25}{:}P \to |(L)|, & p_{26}{:}P \to |(L)||, & p_{27}{:}P \to ||(L)|, & p_{28}{:}P \to ||(L)||, \\
p_{29}{:}Q \to ||(L)K|||, & p_{30}{:}Q \to |||J(L)K||, & & \\
p_{31}{:}R \to |(L)K|||, & p_{32}{:}R \to |||J(L)|, & & \\
p_{33}{:}J \to J|, & p_{34}{:}J \to \epsilon, & & \\
p_{35}{:}K \to K|, & p_{36}{:}K \to \epsilon, & & \\
p_{37}{:}M \to U(L)U(L)N, & & & \\
p_{38}{:}N \to U(L)N, & p_{39}{:}N \to U, & & \\
p_{40}{:}U \to U|, & p_{41}{:}U \to \epsilon. & &
\end{array}
$$

In this grammar, different intermediate symbols have been used to distinguish between different substructures. In fact, the reason why this grammar has so many production rules is that the grammar must be able to distinguish between all the different classes of substructures for which there are different free energy rules according to Turner's thermodynamic model considered in [21].

However, as ϵ-freeness and loop-freeness are required preliminarily, we have to consider another unambiguous SCFG generating the same language $\mathcal{L}$, where we have to guarantee that the same substructures are distinguished as are distinguished in $\mathcal{G}_{\text{sto}}$.

Using the usual way of transforming a non-ϵ-free grammar into an ϵ-free one, the following definition can immediately be obtained from the previous one:

Definition App-.12. The unambigous and ϵ-free SCFG $\mathcal{G}'_{\text{sto}}$ generating exactly the language $\mathcal{L}$ is given by $\mathcal{G}'_{sto} = (I_{\mathcal{G}'_{sto}}, \Sigma_{\mathcal{G}'_{sto}}, R_{\mathcal{G}'_{sto}}, S')$, where

$$I_{\mathcal{G}'_{sto}} = \{S', S, T, C, A, L, G, B, F, H, P, Q, R, J, K, M, N, U\},$$

$\Sigma_{\mathcal{G}'_{\text{sto}}} = \{(,), |\}$ and $\mathcal{R}_{\mathcal{G}'_{sto}}$ contains exactly the following rules:

$$
\begin{array}{llll}
p'_0{:}S' \to S, & & & \\
p'_1{:}S \to A, & p'_2{:}S \to AC, & p'_3{:}S \to TA, & p'_4{:}S \to TAC, \\
p'_5{:}T \to A, & p'_6{:}T \to AC, & p'_7{:}T \to TA, & p'_8{:}T \to TAC, \\
p'_9{:}T \to C, & & & \\
p'_{10}{:}C \to |, & p'_{11}{:}C \to C|, & & \\
p'_{12}{:}A \to (L), & & & \\
p'_{13}{:}L \to (L), & p'_{14}{:}L \to M, & p'_{15}{:}L \to P, & p'_{16}{:}L \to Q, \\
p'_{17}{:}L \to R, & p'_{18}{:}L \to F, & p'_{19}{:}L \to G, & \\
p'_{20}{:}G \to (L)|, & p'_{21}{:}G \to (L)||, & p'_{22}{:}G \to (L)B||, & \\
p'_{23}{:}G \to |(L), & p'_{24}{:}G \to ||(L), & p'_{25}{:}G \to ||B(L), & \\
p'_{26}{:}B \to |, & p'_{27}{:}B \to B|, & & \\
p'_{28}{:}F \to |||, & p'_{29}{:}F \to ||||, & p'_{30}{:}F \to |||||, & p'_{31}{:}F \to |||||H, \\
p'_{32}{:}H \to |, & p'_{33}{:}H \to H|, & & \\
p'_{34}{:}P \to |(L)|, & p'_{35}{:}P \to |(L)||, & p'_{36}{:}P \to ||(L)|, & p'_{37}{:}P \to ||(L)||, \\
p'_{38}{:}Q \to ||(L)|||, & p'_{39}{:}Q \to ||(L)K|||, & p'_{40}{:}Q \to |||(L)||, & p'_{41}{:}Q \to |||J(L)||, \\
p'_{42}{:}Q \to |||(L)K||, & p'_{43}{:}Q \to |||J(L)K||, & & \\
p'_{44}{:}R \to |(L)|||, & p'_{45}{:}R \to |(L)K|||, & p'_{46}{:}R \to |||(L)|, & p'_{47}{:}R \to |||J(L)|, \\
p'_{48}{:}J \to |, & p'_{49}{:}J \to J|, & & \\
p'_{50}{:}K \to |, & p'_{51}{:}K \to K|, & & \\
p'_{52}{:}M \to (L)(L), & p'_{53}{:}M \to U(L)(L), & p'_{54}{:}M \to (L)U(L), & p'_{55}{:}M \to (L)(L)N, \\
p'_{56}{:}M \to U(L)U(L), & p'_{57}{:}M \to U(L)(L)N, & p'_{58}{:}M \to (L)U(L)N, & p'_{59}{:}M \to U(L)U(L)N, \\
p'_{60}{:}N \to (L), & p'_{61}{:}N \to U(L), & p'_{62}{:}N \to (L)N, & p'_{63}{:}N \to U(L)N, \\
p'_{64}{:}N \to U, & & & \\
p'_{65}{:}U \to |, & p'_{66}{:}U \to U|. & &
\end{array}
$$

Unfortunately, the set of productions of $\mathcal{G}'_{\text{sto}}$ contains productions with up to 5 non-terminal symbols in the conclusion. This is not acceptable for our purpose, for the following reason: the desired unranking algorithm makes use of the size of combinatorial classes whose representations somehow are derived from CFGs with particular integer weights on their productions. If we constructed this WCFG by starting with the grammar $\mathcal{G}'_{\text{sto}}$, then this would yield a huge number of production rules. Consequently, the translation would imply a huge specification of the combinatorial classes and the corresponding function to compute their sizes and thus the corresponding unranking algorithm would have to distinguish between an unnecessarily and most importantly unacceptably large number of cases.

Nevertheless, the size of the production set of the weighted grammar underlying the desired unranking algorithm can be significantly reduced by starting with a modification of grammar $\mathcal{G}'_{\text{sto}}$ which has only production rules with minimum possible numbers of non-terminal symbols in the conclusion. In fact, by transforming $\mathcal{G}'_{\text{sto}}$ appropriately considering this observation, we obtained the SCFG $\hat{\mathcal{G}}_{sto}$:

Definition App-.13. The unambiguous ϵ-free SCFG $\hat{\mathcal{G}}_{sto}$ generating exactly the language $\mathcal{L}$ is given by $\hat{\mathcal{G}}_{sto} = (I_{\hat{\mathcal{G}}_{sto}}, \Sigma_{\hat{\mathcal{G}}_{sto}}, R_{\hat{\mathcal{G}}_{sto}}, S')$, where

$$I_{\hat{\mathcal{G}}_{sto}} = \{S', E, S, T, C, A, L, G, D, B, F, H, P, Q, R, V, W, O, J, K, M, X, Y, Z, N, U\},$$

$\Sigma_{\hat{\mathcal{G}}_{sto}} = \{(,), |\}$ and $R_{\hat{\mathcal{G}}_{sto}}$ contains exactly the following rules:

$$\begin{array}{llll}
\widehat{p}_1{:}S' \to E, & & & \\
\widehat{p}_2{:}E \to S, & \widehat{p}_3{:}E \to SC, & & \\
\widehat{p}_4{:}S \to A, & \widehat{p}_5{:}S \to TA, & & \\
\widehat{p}_6{:}T \to E, & \widehat{p}_7{:}T \to C, & & \\
\widehat{p}_8{:}C \to |, & \widehat{p}_9{:}C \to C|, & & \\
\widehat{p}_{10}{:}A \to (L), & & & \\
\widehat{p}_{11}{:}L \to A, & \widehat{p}_{12}{:}L \to M, & \widehat{p}_{13}{:}L \to P, & \widehat{p}_{14}{:}L \to Q, \\
\widehat{p}_{15}{:}L \to R, & \widehat{p}_{16}{:}L \to F, & \widehat{p}_{17}{:}L \to G, & \\
\widehat{p}_{18}{:}G \to A|, & \widehat{p}_{19}{:}G \to AD, & \widehat{p}_{20}{:}G \to |A, & \widehat{p}_{21}{:}G \to DA, \\
\widehat{p}_{22}{:}D \to B|, & & & \\
\widehat{p}_{23}{:}B \to |, & \widehat{p}_{24}{:}B \to B|, & & \\
\widehat{p}_{25}{:}F \to |||, & \widehat{p}_{26}{:}F \to ||||, & \widehat{p}_{27}{:}F \to ||||H, & \\
\widehat{p}_{28}{:}H \to |, & \widehat{p}_{29}{:}H \to H|, & & \\
\widehat{p}_{30}{:}P \to |A|, & \widehat{p}_{31}{:}P \to |A||, & \widehat{p}_{32}{:}P \to ||A|, & \widehat{p}_{33}{:}P \to ||A||, \\
\widehat{p}_{34}{:}Q \to ||O||, & \widehat{p}_{35}{:}Q \to ||V|, & & \\
\widehat{p}_{36}{:}R \to |O||, & \widehat{p}_{37}{:}R \to ||W|, & & \\
\widehat{p}_{38}{:}V \to JO, & & & \\
\widehat{p}_{39}{:}W \to JA, & & & \\
\widehat{p}_{40}{:}O \to AK, & & & \\
\widehat{p}_{41}{:}J \to |, & \widehat{p}_{42}{:}J \to J|, & & \\
\widehat{p}_{43}{:}K \to |, & \widehat{p}_{44}{:}K \to K|, & & \\
\widehat{p}_{45}{:}M \to XY, & & & \\
\widehat{p}_{46}{:}X \to A, & \widehat{p}_{47}{:}X \to UA, & & \\
\widehat{p}_{48}{:}Y \to Z, & & & \\
\widehat{p}_{49}{:}Z \to X, & \widehat{p}_{50}{:}Z \to XN, & & \\
\widehat{p}_{51}{:}N \to Z, & \widehat{p}_{52}{:}N \to U, & & \\
\widehat{p}_{53}{:}U \to |, & \widehat{p}_{54}{:}U \to U|. & &
\end{array}$$

Transforming our SCFG into RNF

Now, we can construct the desired weighted grammar that will be underlying our unranking algorithm: In the first step, we gather all possible chains of productions that do not lengthen the sentential form. In fact, we have to consider all rules $A \to \alpha$, $A \neq S'$, with $|\alpha| = 1$, to obtain all such chains (note that these rules will be removed after step 1). Hence, we have to consider the following set R^1_{mf} of 22 production rules:

$$\begin{array}{l}
\widehat{p}_2{:}E \to S,\\
\widehat{p}_4{:}S \to A,\\
\widehat{p}_6{:}T \to E,\quad \widehat{p}_7{:}T \to C,\\
\widehat{p}_8{:}C \to \mathbf{|},\\
\widehat{p}_{11}{:}L \to A,\ \widehat{p}_{12}{:}L \to M,\ \widehat{p}_{13}{:}L \to P,\ \widehat{p}_{14}{:}L \to Q,\\
\widehat{p}_{15}{:}L \to R,\ \widehat{p}_{16}{:}L \to F,\ \ \widehat{p}_{17}{:}L \to G,\\
\widehat{p}_{23}{:}B \to \mathbf{|},\\
\widehat{p}_{28}{:}H \to \mathbf{|},\\
\widehat{p}_{41}{:}J \to \mathbf{|},\\
\widehat{p}_{43}{:}K \to \mathbf{|},\\
\widehat{p}_{46}{:}X \to A,\\
\widehat{p}_{48}{:}Y \to Z,\\
\widehat{p}_{49}{:}Z \to X,\\
\widehat{p}_{51}{:}N \to Z, \widehat{p}_{52}{:}N \to U,\\
\widehat{p}_{53}{:}U \to \mathbf{|}.
\end{array}$$

Thus, the following 32 chains are gathered in step 1:

$$\begin{array}{lll}
E \Rightarrow S, & targets[E] = \{ & (S, \lambda_{E,S} := \widehat{p}_2, \epsilon),\\
E \Rightarrow S \Rightarrow A, & & (A, \lambda_{E,A} := \widehat{p}_2 \cdot \widehat{p}_4, S)\},\\
 & & \\
S \Rightarrow A, & targets[S] = \{ & (A, \lambda_{S,A} := \widehat{p}_4, \epsilon)\},\\
 & & \\
T \Rightarrow E, & targets[T] = \{ & (E, \lambda_{T,E} := \widehat{p}_6, \epsilon),\\
T \Rightarrow C, & & (C, \lambda_{T,C} := \widehat{p}_7, \epsilon),\\
T \Rightarrow C \Rightarrow \mathbf{|}, & & (\mathbf{|}, \lambda_{T,\mathbf{|}} := \widehat{p}_7 \cdot \widehat{p}_8, C),\\
T \Rightarrow E \Rightarrow S, & & (S, \lambda_{T,S} := \widehat{p}_6 \cdot \widehat{p}_2, E)\},\\
T \Rightarrow E \Rightarrow S \Rightarrow A, & & (A, \lambda_{T,A} := \widehat{p}_6 \cdot \widehat{p}_2 \cdot \widehat{p}_4, ES)\},\\
 & & \\
C \Rightarrow \mathbf{|}, & targets[C] = \{ & (\mathbf{|}, \lambda_{C,\mathbf{|}} := \widehat{p}_8, \epsilon)\},\\
 & & \\
L \Rightarrow A, & targets[L] = \{ & (A, \lambda_{L,A} := \widehat{p}_{11}, \epsilon),\\
L \Rightarrow M, & & (M, \lambda_{L,M} := \widehat{p}_{12}, \epsilon),\\
L \Rightarrow P, & & (P, \lambda_{L,P} := \widehat{p}_{13}, \epsilon),\\
L \Rightarrow Q, & & (Q, \lambda_{L,Q} := \widehat{p}_{14}, \epsilon),\\
L \Rightarrow R, & & (R, \lambda_{L,R} := \widehat{p}_{15}, \epsilon),\\
L \Rightarrow F, & & (F, \lambda_{L,F} := \widehat{p}_{16}, \epsilon),\\
L \Rightarrow G, & & (G, \lambda_{L,G} := \widehat{p}_{17}, \epsilon)\},\\
 & & \\
B \Rightarrow \mathbf{|}, & targets[B] = \{ & (\mathbf{|}, \lambda_{B,\mathbf{|}} := \widehat{p}_{23}, \epsilon)\},\\
 & & \\
H \Rightarrow \mathbf{|}, & targets[H] = \{ & (\mathbf{|}, \lambda_{H,\mathbf{|}} := \widehat{p}_{28}, \epsilon)\},\\
 & & \\
J \Rightarrow \mathbf{|}, & targets[J] = \{ & (\mathbf{|}, \lambda_{J,\mathbf{|}} := \widehat{p}_{41}, \epsilon)\},\\
 & & \\
K \Rightarrow \mathbf{|}, & targets[K] = \{ & (\mathbf{|}, \lambda_{K,\mathbf{|}} := \widehat{p}_{43}, \epsilon)\},\\
 & & \\
X \Rightarrow A, & targets[X] = \{ & (A, \lambda_{X,A} := \widehat{p}_{46}, \epsilon)\},\\
 & & \\
Y \Rightarrow Z, & targets[Y] = \{ & (Z, \lambda_{Y,Z} := \widehat{p}_{48}, \epsilon),\\
Y \Rightarrow Z \Rightarrow X, & & (X, \lambda_{Y,X} := \widehat{p}_{48} \cdot \widehat{p}_{49}, Z),\\
Y \Rightarrow Z \Rightarrow X \Rightarrow A, & & (A, \lambda_{Y,A} := \widehat{p}_{48} \cdot \widehat{p}_{49} \cdot \widehat{p}_{46}, ZX)\},\\
 & & \\
Z \Rightarrow X, & targets[Z] = \{ & (X, \lambda_{Z,X} := \widehat{p}_{49}, \epsilon),\\
Z \Rightarrow X \Rightarrow A, & & (A, \lambda_{Z,A} := \widehat{p}_{49} \cdot \widehat{p}_{46}, X)\},\\
 & & \\
N \Rightarrow Z, & targets[N] = \{ & (Z, \lambda_{N,Z} := \widehat{p}_{51}, \epsilon),\\
N \Rightarrow U, & & (U, \lambda_{N,U} := \widehat{p}_{52}, \epsilon),\\
N \Rightarrow U \Rightarrow \mathbf{|}, & & (\mathbf{|}, \lambda_{N,\mathbf{|}} := \widehat{p}_{52} \cdot \widehat{p}_{53}, U),\\
N \Rightarrow Z \Rightarrow X, & & (X, \lambda_{N,X} := \widehat{p}_{51} \cdot \widehat{p}_{49}, Z),\\
N \Rightarrow Z \Rightarrow X \Rightarrow A, & & (A, \lambda_{N,A} := \widehat{p}_{51} \cdot \widehat{p}_{49} \cdot \widehat{p}_{46}, ZX)\},\\
 & & \\
U \Rightarrow \mathbf{|}, & targets[U] = \{ & (\mathbf{|}, \lambda_{U,\mathbf{|}} := \widehat{p}_{53}, \epsilon)\}.
\end{array}$$

Furthermore, the 22 production rules contained in R^1_{rnf} are now removed. This results in the following set $R^1_{\hat{\mathcal{G}}_{\text{sto}}} := R_{\hat{\mathcal{G}}_{\text{sto}}} \backslash R^1_{rnf}$ of 32 rules:

$$\begin{array}{llll}
\widehat{p}_1{:}S' \to E, & & & \\
\widehat{p}_3{:}E \to SC, & & & \\
\widehat{p}_5{:}S \to TA, & & & \\
\widehat{p}_9{:}C \to C|, & & & \\
\widehat{p}_{10}{:}A \to (L), & & & \\
\widehat{p}_{18}{:}G \to A|, & \widehat{p}_{19}{:}G \to AD, & \widehat{p}_{20}{:}G \to |A, & \widehat{p}_{21}{:}G \to DA, \\
\widehat{p}_{22}{:}D \to B|, & & & \\
\widehat{p}_{24}{:}B \to B|, & & & \\
\widehat{p}_{25}{:}F \to |||, & \widehat{p}_{26}{:}F \to ||||, & \widehat{p}_{27}{:}F \to ||||H, & \\
\widehat{p}_{29}{:}H \to H|, & & & \\
\widehat{p}_{30}{:}P \to |A|, & \widehat{p}_{31}{:}P \to |A||, & \widehat{p}_{32}{:}P \to ||A|, & \widehat{p}_{33}{:}P \to ||A||, \\
\widehat{p}_{34}{:}Q \to ||O||, & \widehat{p}_{35}{:}Q \to ||V|, & & \\
\widehat{p}_{36}{:}R \to |O||, & \widehat{p}_{37}{:}R \to ||W|, & & \\
\widehat{p}_{38}{:}V \to JO, & & & \\
\widehat{p}_{39}{:}W \to JA, & & & \\
\widehat{p}_{40}{:}O \to AK, & & & \\
\widehat{p}_{42}{:}J \to J|, & & & \\
\widehat{p}_{44}{:}K \to K|, & & & \\
\widehat{p}_{45}{:}M \to XY, & & & \\
\widehat{p}_{47}{:}X \to UA, & & & \\
\widehat{p}_{50}{:}Z \to XN, & & & \\
\widehat{p}_{54}{:}U \to U|. & & &
\end{array}$$

Additionally, in step 2, for each chain a new intermediate symbol and a new production are introduced. Thus, according to the 32 chains gathered in step 1, we here obtain the following set R^2_{rnf} of 32 new production rules:

$$\begin{array}{llll}
1{:}E^{S,\epsilon} \to S, & 1{:}E^{A,S} \to A, & & \\
1{:}S^{A,\epsilon} \to A, & & & \\
1{:}T^{E,\epsilon} \to E, & 1{:}T^{C,\epsilon} \to C, & 1{:}T^{|,C} \to |, & \\
1{:}T^{S,E} \to S, & 1{:}T^{A,ES} \to A, & & \\
1{:}C^{|,\epsilon} \to |, & & & \\
1{:}L^{A,\epsilon} \to A, & 1{:}L^{M,\epsilon} \to M, & 1{:}L^{P,\epsilon} \to P, & 1{:}L^{Q,\epsilon} \to Q, \\
1{:}L^{R,\epsilon} \to R, & 1{:}L^{F,\epsilon} \to F, & 1{:}L^{G,\epsilon} \to G, & \\
1{:}B^{|,\epsilon} \to |, & & & \\
1{:}H^{|,\epsilon} \to |, & & & \\
1{:}J^{|,\epsilon} \to |, & & & \\
1{:}K^{|,\epsilon} \to |, & & & \\
1{:}X^{A,\epsilon} \to A, & & & \\
1{:}Y^{Z,\epsilon} \to Z, & 1{:}Y^{X,Z} \to X, & 1{:}Y^{A,ZX} \to A, & \\
1{:}Z^{X,\epsilon} \to X, & & & \\
1{:}Z^{A,X} \to A, & & & \\
1{:}N^{Z,\epsilon} \to Z, & 1{:}N^{U,\epsilon} \to U, & 1{:}N^{|,U} \to |, & \\
1{:}N^{X,Z} \to X, & 1{:}N^{A,ZX} \to A, & & \\
1{:}U^{|,\epsilon} \to |. & & &
\end{array}$$

In step 3, for each occurrence of a non-terminal symbol in the conclusion of a production and each chain starting with this non-terminal symbol, we have to add a new production with the corresponding new intermediate symbol instead of the considered one. Thus, in step 3, the remaining 32 production rules from $R^1_{\hat{\mathcal{G}}_{\text{sto}}} := R_{\hat{\mathcal{G}}_{\text{sto}}} \backslash R^1_{rmf}$ are transformed (according to R^2_{rmf}) into the following set $R^2_{\hat{\mathcal{G}}_{\text{sto}}}$ of 79 new rules:

$$\begin{array}{rr}
\widehat{p}_1{:}S' \to E, & \widehat{p}_1 \cdot \lambda_{E,S}{:}S' \to E^{S,\epsilon}, \\
\widehat{p}_1 \cdot \lambda_{E,A}{:}S' \to E^{A,S}, & \\
\widehat{p}_3{:}E \to SC, & \widehat{p}_3 \cdot \lambda_{S,A}{:}E \to S^{A,\epsilon}C, \\
\widehat{p}_3 \cdot \lambda_{C,|}{:}E \to SC^{|,\epsilon}, & \widehat{p}_3 \cdot \lambda_{S,A} \cdot \lambda_{C,|}{:}E \to S^{A,\epsilon}C^{|,\epsilon}, \\
\widehat{p}_5{:}S \to TA, & \widehat{p}_5 \cdot \lambda_{T,E}{:}S \to T^{E,\epsilon}A, \\
\widehat{p}_5 \cdot \lambda_{T,C}{:}S \to T^{C,\epsilon}A, & \widehat{p}_5 \cdot \lambda_{T,|}{:}S \to T^{|,C}A, \\
\widehat{p}_5 \cdot \lambda_{T,S}{:}S \to T^{S,E}A, & \widehat{p}_5 \cdot \lambda_{T,A}{:}S \to T^{A,ES}A, \\
\widehat{p}_9{:}C \to C|, & \widehat{p}_9 \cdot \lambda_{C,|}{:}C \to C^{|,\epsilon}|, \\
\widehat{p}_{10}{:}A \to (L), & \widehat{p}_{10} \cdot \lambda_{L,A}{:}A \to (L^{A,\epsilon}), \\
\widehat{p}_{10} \cdot \lambda_{L,M}{:}A \to (L^{M,\epsilon}), & \widehat{p}_{10} \cdot \lambda_{L,P}{:}A \to (L^{P,\epsilon}), \\
\widehat{p}_{10} \cdot \lambda_{L,Q}{:}A \to (L^{Q,\epsilon}), & \widehat{p}_{10} \cdot \lambda_{L,R}{:}A \to (L^{R,\epsilon}), \\
\widehat{p}_{10} \cdot \lambda_{L,F}{:}A \to (L^{F,\epsilon}), & \widehat{p}_{10} \cdot \lambda_{L,G}{:}A \to (L^{G,\epsilon}), \\
\widehat{p}_{18}{:}G \to A|, & \widehat{p}_{19}{:}G \to AD, \\
\widehat{p}_{20}{:}G \to |A, & \widehat{p}_{21}{:}G \to DA, \\
\widehat{p}_{22}{:}D \to B|, & \widehat{p}_{22} \cdot \lambda_{B,|}{:}D \to B^{|,\epsilon}|, \\
\widehat{p}_{24}{:}B \to B|, & \widehat{p}_{24} \cdot \lambda_{B,|}{:}B \to B^{|,\epsilon}|, \\
\widehat{p}_{25}{:}F \to |||, & \widehat{p}_{26}{:}F \to ||||, \\
\widehat{p}_{27}{:}F \to ||||H, & \widehat{p}_{27} \cdot \lambda_{H,|}{:}F \to ||||H^{|,\epsilon}, \\
\widehat{p}_{29}{:}H \to H|, & \widehat{p}_{29} \cdot \lambda_{H,|}{:}H \to H^{|,\epsilon}|, \\
\widehat{p}_{30}{:}P \to |A|, & \widehat{p}_{31}{:}P \to |A||, \\
\widehat{p}_{32}{:}P \to ||A|, & \widehat{p}_{33}{:}P \to ||A||, \\
\widehat{p}_{34}{:}Q \to ||O||, & \widehat{p}_{35}{:}Q \to ||V|, \\
\widehat{p}_{36}{:}R \to |O||, & \widehat{p}_{37}{:}R \to ||W|, \\
\widehat{p}_{38}{:}V \to JO, & \widehat{p}_{38} \cdot \lambda_{J,|}{:}V \to J^{|,\epsilon}O, \\
\widehat{p}_{39}{:}W \to JA, & \widehat{p}_{39} \cdot \lambda_{J,|}{:}W \to J^{|,\epsilon}A, \\
\widehat{p}_{40}{:}O \to AK, & \widehat{p}_{40} \cdot \lambda_{K,|}{:}O \to AK^{|,\epsilon}, \\
\widehat{p}_{42}{:}J \to J|, & \widehat{p}_{42} \cdot \lambda_{J,|}{:}J \to J^{|,\epsilon}|, \\
\widehat{p}_{44}{:}K \to K|, & \widehat{p}_{44} \cdot \lambda_{K,|}{:}K \to K^{|,\epsilon}|, \\
\widehat{p}_{45}{:}M \to XY, & \widehat{p}_{45} \cdot \lambda_{Y,Z}{:}M \to XY^{Z,\epsilon}, \\
\widehat{p}_{45} \cdot \lambda_{Y,X}{:}M \to XY^{X,Z}, & \widehat{p}_{45} \cdot \lambda_{Y,A}{:}M \to XY^{A,ZX}, \\
\widehat{p}_{45} \cdot \lambda_{X,A}{:}M \to X^{A,\epsilon}Y, & \widehat{p}_{45} \cdot \lambda_{X,A} \cdot \lambda_{Y,Z}{:}M \to X^{A,\epsilon}Y^{Z,\epsilon}, \\
\widehat{p}_{45} \cdot \lambda_{X,A} \cdot \lambda_{Y,X}{:}M \to X^{A,\epsilon}Y^{X,Z}, & \widehat{p}_{45} \cdot \lambda_{X,A} \cdot \lambda_{Y,A}{:}M \to X^{A,\epsilon}Y^{A,ZX}, \\
\widehat{p}_{47}{:}X \to UA, & \widehat{p}_{47} \cdot \lambda_{U,|}{:}X \to U^{|,\epsilon}A, \\
\widehat{p}_{50}{:}Z \to XN, & \widehat{p}_{50} \cdot \lambda_{N,Z}{:}Z \to XN^{Z,\epsilon}, \\
\widehat{p}_{50} \cdot \lambda_{N,U}{:}Z \to XN^{U,\epsilon}, & \widehat{p}_{50} \cdot \lambda_{N,|}{:}Z \to XN^{|,U}, \\
\widehat{p}_{50} \cdot \lambda_{N,X}{:}Z \to XN^{X,Z}, & \widehat{p}_{50} \cdot \lambda_{N,A}{:}Z \to XN^{A,ZX}, \\
\widehat{p}_{50} \cdot \lambda_{X,A}{:}Z \to X^{A,\epsilon}N, & \widehat{p}_{50} \cdot \lambda_{X,A} \cdot \lambda_{N,Z}{:}Z \to X^{A,\epsilon}N^{Z,\epsilon}, \\
\widehat{p}_{50} \cdot \lambda_{X,A} \cdot \lambda_{N,U}{:}Z \to X^{A,\epsilon}N^{U,\epsilon}, & \widehat{p}_{50} \cdot \lambda_{X,A} \cdot \lambda_{N,|}{:}Z \to X^{A,\epsilon}N^{|,U}, \\
\widehat{p}_{50} \cdot \lambda_{X,A} \cdot \lambda_{N,X}{:}Z \to X^{A,\epsilon}N^{X,Z}, & \widehat{p}_{50} \cdot \lambda_{X,A} \cdot \lambda_{N,A}{:}Z \to X^{A,\epsilon}N^{A,ZX}, \\
\widehat{p}_{54}{:}U \to U|, & \widehat{p}_{54} \cdot \lambda_{U,|}{:}U \to U^{|,\epsilon}|.
\end{array}$$

In step 4, we must delete all intermediate symbols that no longer occur as premise. Obviously, intermediate symbols no longer occurring as premise of a production are

$$T, L, N, Y.$$

We easily observe that the productions that contain at least one of these 4 intermediate symbols in the conclusion and thus have to be removed are exactly the following ones:

$$\begin{array}{l}
\widehat{p}_5{:}S \to TA,\\
\widehat{p}_{10}{:}A \to (L),\\
\widehat{p}_{45}{:}M \to XY,\ \widehat{p}_{45}\cdot\lambda_{X,A}{:}M \to X^{A,\epsilon}Y,\\
\widehat{p}_{50}{:}Z \to XN,\ \widehat{p}_{50}\cdot\lambda_{X,A}{:}Z \to X^{A,\epsilon}N.
\end{array}$$

Consequently, after the removal of these 6 rules from $R^2_{\hat{\mathcal{G}}_{\text{sto}}}$, there still remain 73 new production rules.

Finally in step 5, we must make sure that the conclusion of all productions with premise *S'* (axiom of $\hat{\mathcal{G}}_{sto}$ that we started with) does not have a length greater than 1. However, since there is only one production with premise *S'* in our start grammar $\hat{\mathcal{G}}_{sto}$ and the conclusion of this production has size 1, there is nothing to do. Thus, the resulting new grammar is given by:

Definition App-.14. The WCFG $\hat{\mathcal{G}}^*_{sto}$ generating exactly the language $\mathcal{L}$ is given by $\hat{\mathcal{G}}^*_{\text{sto}} = (I_{\hat{\mathcal{G}}^*_{\text{sto}}} \cup I'_{\hat{\mathcal{G}}^*_{\text{sto}}}, \Sigma_{\hat{\mathcal{G}}^*_{\text{sto}}}, R_{\hat{\mathcal{G}}^*_{\text{sto}}} \cup R'_{\hat{\mathcal{G}}^*_{\text{sto}}}, S')$, where

$$I_{\hat{\mathcal{G}}^*_{sto}} = \{S', E, S, C, A, G, D, B, F, H, P, Q, R, V, W, O, J, K, M, X, Z, U\},$$

$$\begin{aligned}
I'_{\hat{\mathcal{G}}^*_{\text{sto}}} = \{&E^{S,\epsilon}, E^{A,S}, S^{A,\epsilon},\\
&T^{E,\epsilon}, T^{C,\epsilon}, T^{|,C}, T^{S,E}, T^{A,ES}, C^{|,\epsilon},\\
&L^{A,\epsilon}, L^{M,\epsilon}, L^{P,\epsilon}, L^{Q,\epsilon}, L^{R,\epsilon}, L^{F,\epsilon}, L^{G,\epsilon},\\
&B^{|,\epsilon}, H^{|,\epsilon}, J^{|,\epsilon}, K^{|,\epsilon},\\
&X^{A,\epsilon}, Y^{Z,\epsilon}, Y^{X,Z}, Y^{A,ZX}, Z^{X,\epsilon}, Z^{A,X},\\
&N^{Z,\epsilon}, N^{U,\epsilon}, N^{|,U}, N^{X,Z}, N^{A,ZX}, U^{|,\epsilon}\},
\end{aligned}$$

$\Sigma_{\hat{\mathcal{G}}^*_{sto}} = \{(,), |\}$ and $R_{\hat{\mathcal{G}}^*_{sto}}$ contains exactly the following rules:

$$\begin{array}{llll}
\lambda_1{:}S' \to E, & \lambda_2{:}S' \to E^{S,\epsilon}, & \lambda_3{:}S' \to E^{A,S}, & \\
\lambda_4{:}E \to SC, & \lambda_5{:}E \to S^{A,\epsilon}C, & \lambda_6{:}E \to SC^{|,\epsilon}, & \lambda_7{:}E \to S^{A,\epsilon}C^{|,\epsilon},\\
\lambda_8{:}S \to T^{E,\epsilon}A, & \lambda_9{:}S \to T^{C,\epsilon}A, & \lambda_{10}{:}S \to T^{|,C}A, & \\
\lambda_{11}{:}S \to T^{S,E}A, & \lambda_{12}{:}S \to T^{A,ES}A, & & \\
\lambda_{13}{:}C \to C|, & \lambda_{14}{:}C \to C^{|,\epsilon}|, & & \\
\lambda_{15}{:}A \to (L^{A,\epsilon}), & \lambda_{16}{:}A \to (L^{M,\epsilon}), & \lambda_{17}{:}A \to (L^{P,\epsilon}), & \lambda_{18}{:}A \to (L^{Q,\epsilon}),\\
\lambda_{19}{:}A \to (L^{R,\epsilon}), & \lambda_{20}{:}A \to (L^{F,\epsilon}), & \lambda_{21}{:}A \to (L^{G,\epsilon}), & \\
\lambda_{22}{:}G \to A|, & \lambda_{23}{:}G \to AD, & \lambda_{24}{:}G \to |A, & \lambda_{25}{:}G \to DA,\\
\lambda_{26}{:}D \to B|, & \lambda_{27}{:}D \to B^{|,\epsilon}|, & & \\
\lambda_{28}{:}B \to B|, & \lambda_{29}{:}B \to B^{|,\epsilon}|, & & \\
\lambda_{30}{:}F \to |||, & \lambda_{31}{:}F \to ||||, & \lambda_{32}{:}F \to ||||H, & \lambda_{33}{:}F \to ||||H^{|,\epsilon},\\
\lambda_{34}{:}H \to H|, & \lambda_{35}{:}H \to H^{|,\epsilon}|, & & \\
\lambda_{36}{:}P \to |A|, & \lambda_{37}{:}P \to |A||, & \lambda_{38}{:}P \to ||A|, & \lambda_{39}{:}P \to ||A||,\\
\lambda_{40}{:}Q \to ||O||, & \lambda_{41}{:}Q \to ||V|, & & \\
\lambda_{42}{:}R \to |O||, & \lambda_{43}{:}R \to ||W|, & & \\
\lambda_{44}{:}V \to JO, & \lambda_{45}{:}V \to J^{|,\epsilon}O, & & \\
\lambda_{46}{:}W \to JA, & \lambda_{47}{:}W \to J^{|,\epsilon}A, & & \\
\lambda_{48}{:}O \to AK, & \lambda_{49}{:}O \to AK^{|,\epsilon}, & & \\
\lambda_{50}{:}J \to J|, & \lambda_{51}{:}J \to J^{|,\epsilon}|, & & \\
\lambda_{52}{:}K \to K|, & \lambda_{53}{:}K \to K^{|,\epsilon}|, & & \\
\lambda_{54}{:}M \to XY^{Z,\epsilon}, & \lambda_{55}{:}M \to XY^{X,Z}, & \lambda_{56}{:}M \to XY^{A,ZX}, & \\
\lambda_{57}{:}M \to X^{A,\epsilon}Y^{Z,\epsilon}, & \lambda_{58}{:}M \to X^{A,\epsilon}Y^{X,Z}, & \lambda_{59}{:}M \to X^{A,\epsilon}Y^{A,ZX}, & \\
\lambda_{60}{:}X \to UA, & \lambda_{61}{:}X \to U^{|,\epsilon}A, & & \\
\lambda_{62}{:}Z \to XN^{Z,\epsilon}, & \lambda_{63}{:}Z \to XN^{U,\epsilon}, & \lambda_{64}{:}Z \to XN^{|,U}, & \\
\lambda_{65}{:}Z \to XN^{X,Z}, & \lambda_{66}{:}Z \to XN^{A,ZX}, & & \\
\lambda_{67}{:}Z \to X^{A,\epsilon}N^{Z,\epsilon}, & \lambda_{68}{:}Z \to X^{A,\epsilon}N^{U,\epsilon}, & \lambda_{69}{:}Z \to X^{A,\epsilon}N^{|,U}, & \\
\lambda_{70}{:}Z \to X^{A,\epsilon}N^{X,Z}, & \lambda_{71}{:}Z \to X^{A,\epsilon}N^{A,ZX}, & & \\
\lambda_{72}{:}U \to U|, & \lambda_{73}{:}U \to U^{|,\epsilon}|, & &
\end{array}$$

whereas $R'_{\hat{\mathcal{G}}^*_{sto}}$ contains exactly the following rules:

$$\begin{array}{llll}
\lambda_{74}{:}E^{S,\epsilon} \to S, & \lambda_{75}{:}E^{A,S} \to A, & & \\
\lambda_{76}{:}S^{A,\epsilon} \to A, & & & \\
\lambda_{77}{:}T^{E,\epsilon} \to E, & \lambda_{78}{:}T^{C,\epsilon} \to C, & \lambda_{79}{:}T^{|,C} \to |, & \\
\lambda_{80}{:}T^{S,E} \to S, & \lambda_{81}{:}T^{A,ES} \to A, & & \\
\lambda_{82}{:}C^{|,\epsilon} \to |, & & & \\
\lambda_{83}{:}L^{A,\epsilon} \to A, & \lambda_{84}{:}L^{M,\epsilon} \to M, & \lambda_{85}{:}L^{P,\epsilon} \to P, & \lambda_{86}{:}L^{Q,\epsilon} \to Q, \\
\lambda_{87}{:}L^{R,\epsilon} \to R, & \lambda_{88}{:}L^{F,\epsilon} \to F, & \lambda_{89}{:}L^{G,\epsilon} \to G, & \\
\lambda_{90}{:}B^{|,\epsilon} \to |, & & & \\
\lambda_{91}{:}H^{|,\epsilon} \to |, & & & \\
\lambda_{92}{:}J^{|,\epsilon} \to |, & & & \\
\lambda_{93}{:}K^{|,\epsilon} \to |, & & & \\
\lambda_{94}{:}X^{A,\epsilon} \to A, & & & \\
\lambda_{95}{:}Y^{Z,\epsilon} \to Z, & \lambda_{96}{:}Y^{X,Z} \to X, & \lambda_{97}{:}Y^{A,ZX} \to A, & \\
\lambda_{98}{:}Z^{X,\epsilon} \to X, & \lambda_{99}{:}Z^{A,X} \to A, & & \\
\lambda_{100}{:}N^{Z,\epsilon} \to Z, & \lambda_{101}{:}N^{U,\epsilon} \to U, & \lambda_{102}{:}N^{|,U} \to |, & \\
\lambda_{103}{:}N^{X,Z} \to X, & \lambda_{104}{:}N^{A,ZX} \to A, & & \\
\lambda_{105}{:}U^{|,\epsilon} \to |. & & &
\end{array}$$

Reweighting the Production Rules

Now, the weights of the 73 production rules given in the subset of productions $R_{\hat{\mathcal{G}}^*_{sto}}$ have to be reweighted. In order to achieve this goal, we first have to compute the two common denominators s and c, where s is the common denominator of the weights of productions with premise S' (i.e., of productions number 1 to 3), and c is the common denominator of the weights of the remaining productions (i.e., of productions number 4 to 73) of $R_{\hat{\mathcal{G}}^*_{sto}}$. Using the rounded probabilities (weights) for the production rules of $\hat{\mathcal{G}}^*_{sto}$ as given in Table 5, we immediately find the smallest common denominators to be s = 10,000 and c = 10,000.

The desired new weights for the considered set of productions $R_{\hat{\mathcal{G}}^*_{sto}}$ are then computed by multiplying the old weights of productions with source S' by s, and by multiplying the old weights of productions $A \to \alpha$, $A \neq S'$ (and $A \in I_{\hat{\mathcal{G}}^*_{\mathrm{sto}}}$), by $c^{|\alpha|-1}$.

Formally, for the reweighted set of productions $R_{\hat{\mathcal{G}}^*_{sto}}$, we get the following weights:

$$\mu_i := \lambda_i \cdot s, \text{ for } i \in \{1, 2, 3\}$$

and

$$\mu_i := \lambda_i \cdot c^{|\alpha_i|-1}, \text{ where } \lambda_i : A_i \to \alpha_i, \text{ for } i \in \{4, ..., 73\}.$$

The resulting integer weights can be found in Table 6.

Transforming Reweighted Grammar into Admissible Specification

Given the reweighted grammar $\hat{\mathcal{G}}^*_{sto}$, we immediately obtain the following admissible specification of the corresponding combinatorial classes (note that this specification has already been simplified by removing classes that are only duplicates of others):

$$\begin{array}{lll}
\mathcal{E}_1 = \mathcal{S} \times \mathcal{C}, & \mathcal{E}_2 = \mathcal{A} \times \mathcal{C}, & \\
\mathcal{E}_3 = \mathcal{S} \times \alpha_|, & \mathcal{E}_4 = \mathcal{A} \times \alpha_|, & \\
\mathcal{S}_1 = \mathcal{E} \times \mathcal{A}, & \mathcal{S}_2 = \mathcal{C} \times \mathcal{A}, & \mathcal{S}_3 = \alpha_| \times \mathcal{A}, \\
\mathcal{S}_4 = \mathcal{S} \times \mathcal{A}, & \mathcal{S}_5 = \mathcal{A} \times \mathcal{A}, & \\
\mathcal{C}_1 = \mathcal{C} \times \alpha_|, & \mathcal{C}_2 = \alpha_| \times \alpha_|, & \\
\mathcal{A}_1 = \alpha_(\times \mathcal{A} \times \alpha_), & \mathcal{A}_2 = \alpha_(\times \mathcal{M} \times \alpha_), & \mathcal{A}_3 = \alpha_(\times \mathcal{P} \times \alpha_), \\
\mathcal{A}_4 = \alpha_(\times \mathcal{Q} \times \alpha_), & \mathcal{A}_5 = \alpha_(\times \mathcal{R} \times \alpha_), & \mathcal{A}_6 = \alpha_(\times \mathcal{F} \times \alpha_), \\
\mathcal{A}_7 = \alpha_(\times \mathcal{G} \times \alpha_), & & \\
\mathcal{G}_1 = \mathcal{A} \times \alpha_|, & \mathcal{G}_2 = \mathcal{A} \times \mathcal{D}, & \\
\mathcal{G}_3 = \alpha_| \times \mathcal{A}, & \mathcal{G}_4 = \mathcal{D} \times \mathcal{A}, & \\
\mathcal{D}_1 = \mathcal{B} \times \alpha_|, & \mathcal{D}_2 = \alpha_| \times \alpha_|, & \\
\mathcal{B}_1 = \mathcal{B} \times \alpha_|, & \mathcal{B}_2 = \alpha_| \times \alpha_|, & \\
\mathcal{F}_1 = \alpha_| \times \alpha_| \times \alpha_|, & \mathcal{F}_2 = \alpha_| \times \alpha_| \times \alpha_| \times \alpha_|, & \mathcal{F}_3 = \alpha_| \times \alpha_| \times \alpha_| \times \alpha_| \times \mathcal{H}, \\
\mathcal{F}_4 = \alpha_| \times \alpha_| \times \alpha_| \times \alpha_| \times \alpha_|, & & \\
\mathcal{H}_1 = \mathcal{H} \times \alpha_|, & \mathcal{H}_2 = \alpha_| \times \alpha_|, & \\
\mathcal{P}_1 = \alpha_| \times \mathcal{A} \times \alpha_|, & \mathcal{P}_2 = \alpha_| \times \mathcal{A} \times \alpha_| \times \alpha_|, & \mathcal{P}_3 = \alpha_| \times \alpha_| \times \mathcal{A} \times \alpha_|, \\
\mathcal{P}_4 = \alpha_| \times \alpha_| \times \mathcal{A} \times \alpha_| \times \alpha_|, & & \\
\mathcal{Q}_1 = \alpha_| \times \alpha_| \times \mathcal{O} \times \alpha_| \times \alpha_|, & \mathcal{Q}_2 = \alpha_| \times \alpha_| \times \mathcal{V} \times \alpha_|, & \\
\mathcal{R}_1 = \alpha_| \times \mathcal{O} \times \alpha_| \times \alpha_|, & \mathcal{R}_2 = \alpha_| \times \alpha_| \times \mathcal{W} \times \alpha_|, & \\
\mathcal{V}_1 = \mathcal{J} \times \mathcal{O}, & \mathcal{V}_2 = \alpha_| \times \mathcal{O}, & \\
\mathcal{W}_1 = \mathcal{J} \times \mathcal{A}, & \mathcal{W}_2 = \alpha_| \times \mathcal{A}, & \\
\mathcal{O}_1 = \mathcal{A} \times \mathcal{K}, & \mathcal{O}_2 = \mathcal{A} \times \alpha_|, & \\
\mathcal{J}_1 = \mathcal{J} \times \alpha_|, & \mathcal{J}_2 = \alpha_| \times \alpha_|, & \\
\mathcal{K}_1 = \mathcal{K} \times \alpha_|, & \mathcal{K}_2 = \alpha_| \times \alpha_|, & \\
\mathcal{M}_1 = \mathcal{X} \times \mathcal{Z}, & \mathcal{M}_2 = \mathcal{X} \times \mathcal{X}, & \mathcal{M}_3 = \mathcal{X} \times \mathcal{A}, \\
\mathcal{M}_4 = \mathcal{A} \times \mathcal{Z}, & \mathcal{M}_5 = \mathcal{A} \times \mathcal{X}, & \mathcal{M}_6 = \mathcal{A} \times \mathcal{A}, \\
\mathcal{X}_1 = \mathcal{U} \times \mathcal{A}, & \mathcal{X}_2 = \alpha_| \times \mathcal{A}, & \\
\mathcal{Z}_1 = \mathcal{X} \times \mathcal{Z}, & \mathcal{Z}_2 = \mathcal{X} \times \mathcal{U}, & \mathcal{Z}_3 = \mathcal{X} \times \alpha_|, \\
\mathcal{Z}_4 = \mathcal{X} \times \mathcal{X}, & \mathcal{Z}_5 = \mathcal{X} \times \mathcal{A}, & \\
\mathcal{Z}_6 = \mathcal{A} \times \mathcal{Z}, & \mathcal{Z}_7 = \mathcal{A} \times \mathcal{U}, & \mathcal{Z}_8 = \mathcal{A} \times \alpha_|, \\
\mathcal{Z}_9 = \mathcal{A} \times \mathcal{X}, & \mathcal{Z}_{10} = \mathcal{A} \times \mathcal{A}, & \\
\mathcal{U}_1 = \mathcal{U} \times \alpha_|, & \mathcal{U}_2 = \alpha_| \times \alpha_|, &
\end{array}$$

$$\begin{array}{l}
\mathcal{S}' = \mu_1 \cdot \mathcal{E} + \mu_2 \cdot \mathcal{S} + \mu_3 \cdot \mathcal{A}, \\
\mathcal{E} = \mu_4 \cdot \mathcal{E}_1 + \mu_5 \cdot \mathcal{E}_2 + \mu_6 \cdot \mathcal{E}_3 + \mu_7 \cdot \mathcal{E}_4, \\
\mathcal{S} = \mu_8 \cdot \mathcal{S}_1 + \mu_9 \cdot \mathcal{S}_2 + \mu_{10} \cdot \mathcal{S}_3 + \mu_{11} \cdot \mathcal{S}_4 + \mu_{12} \cdot \mathcal{S}_5, \\
\mathcal{C} = \mu_{13} \cdot \mathcal{C}_1 + \mu_{14} \cdot \mathcal{C}_2, \\
\mathcal{A} = \mu_{15} \cdot \mathcal{A}_1 + \mu_{16} \cdot \mathcal{A}_2 + \mu_{17} \cdot \mathcal{A}_3 + \mu_{18} \cdot \mathcal{A}_4 + \mu_{19} \cdot \mathcal{A}_5 + \mu_{20} \cdot \mathcal{A}_6 + \mu_{21} \cdot \mathcal{A}_7, \\
\mathcal{G} = \mu_{22} \cdot \mathcal{G}_1 + \mu_{23} \cdot \mathcal{G}_2 + \mu_{24} \cdot \mathcal{G}_3 + \mu_{25} \cdot \mathcal{G}_4, \\
\mathcal{D} = \mu_{26} \cdot \mathcal{D}_1 + \mu_{27} \cdot \mathcal{D}_2, \\
\mathcal{B} = \mu_{28} \cdot \mathcal{B}_1 + \mu_{29} \cdot \mathcal{B}_2, \\
\mathcal{F} = \mu_{30} \cdot \mathcal{F}_1 + \mu_{31} \cdot \mathcal{F}_2 + \mu_{32} \cdot \mathcal{F}_3 + \mu_{33} \cdot \mathcal{F}_4, \\
\mathcal{H} = \mu_{34} \cdot \mathcal{H}_1 + \mu_{35} \cdot \mathcal{H}_2, \\
\mathcal{P} = \mu_{36} \cdot \mathcal{P}_1 + \mu_{37} \cdot \mathcal{P}_2 + \mu_{38} \cdot \mathcal{P}_3 + \mu_{39} \cdot \mathcal{P}_4, \\
\mathcal{Q} = \mu_{40} \cdot \mathcal{Q}_1 + \mu_{41} \cdot \mathcal{Q}_2, \\
\mathcal{R} = \mu_{42} \cdot \mathcal{R}_1 + \mu_{43} \cdot \mathcal{R}_2, \\
\mathcal{V} = \mu_{44} \cdot \mathcal{V}_1 + \mu_{45} \cdot \mathcal{V}_2, \\
\mathcal{W} = \mu_{46} \cdot \mathcal{W}_1 + \mu_{47} \cdot \mathcal{W}_2, \\
\mathcal{O} = \mu_{48} \cdot \mathcal{O}_1 + \mu_{49} \cdot \mathcal{O}_2, \\
\mathcal{J} = \mu_{50} \cdot \mathcal{J}_1 + \mu_{51} \cdot \mathcal{J}_2, \\
\mathcal{K} = \mu_{52} \cdot \mathcal{K}_1 + \mu_{53} \cdot \mathcal{K}_2, \\
\mathcal{M} = \mu_{54} \cdot \mathcal{M}_1 + \mu_{55} \cdot \mathcal{M}_2 + \mu_{56} \cdot \mathcal{M}_3 + \mu_{57} \cdot \mathcal{M}_4 + \mu_{58} \cdot \mathcal{M}_5 + \mu_{59} \cdot \mathcal{M}_6, \\
\mathcal{X} = \mu_{60} \cdot \mathcal{X}_1 + \mu_{61} \cdot \mathcal{X}_2, \\
\mathcal{Z} = \mu_{62} \cdot \mathcal{Z}_1 + \mu_{63} \cdot \mathcal{Z}_2 + \mu_{64} \cdot \mathcal{Z}_3 + \mu_{65} \cdot \mathcal{Z}_4 + \mu_{66} \cdot \mathcal{Z}_5 + \mu_{67} \cdot \mathcal{Z}_6 + \mu_{68} \cdot \mathcal{Z}_7 + \mu_{69} \cdot \mathcal{Z}_8 + \mu_{70} \cdot \mathcal{Z}_9 + \mu_{71} \cdot \mathcal{Z}_{10}, \\
\mathcal{U} = \mu_{72} \cdot \mathcal{U}_1 + \mu_{73} \cdot \mathcal{U}_2.
\end{array}$$

Now, this(simplified) specification can easily be transformed into the following recursive form for the function `size`:

Table 5 Floating point approximations of the probabilities (weights) λ_i, $1 \leq i \leq 73$, for the production rules of the grammar $\hat{\mathcal{G}}^*_{sto}$ (rounded to four decimal places)

Nonterminal *Nt*	Weights of Rules with Premise *Nt*			
S′	λ_1 : = 1.0000,	λ_2 : = 0.0212,	λ_3 : = 0.0003,	
E	λ_4 : = 0.9788,	λ_5 : = 0.0134,	λ_6 : = 0.0944,	λ_7 : = 0.0013,
S	λ_8 : = 0.8559,	λ_9 : = 0.1304,	λ_{10} : = 0.0126,	λ_{11} : = 0.0181,
	λ_{12} : = 0.0002,			
C	λ_{13} : = 0.9036,	λ_{14} : = 0.0871,		
A	λ_{15} : = 0.7630,	λ_{16} : = 0.0402,	λ_{17} : = 0.0186,	λ_{18} : = 0.0367,
	λ_{19} : = 0.0072,	λ_{20} : = 0.0858,	λ_{21} : = 0.0484,	
G	λ_{22} : = 0.3038,	λ_{23} : = 0.1884,	λ_{24} : = 0.3081,	λ_{25} : = 0.1996,
D	λ_{26} : = 1.0000,	λ_{27} : = 0.3896,		
B	λ_{28} : = 0.6104,	λ_{29} : = 0.2378,		
F	λ_{30} : = 0.0575,	λ_{31} : = 0.3409,	λ_{32} : = 0.6016,	λ33: = 0.1211,
H	λ_{34} : = 0.7987,	λ_{35} : = 0.1608,		
P	λ_{36} : = 0.1085,	λ_{37} : = 0.2144,	λ_{38} : = 0.2011,	λ_{39} : = 0.4760,
Q	λ_{40} : = 0.1713,	λ_{41} : = 0.8287,		
R	λ_{42} : = 0.4150,	λ_{43} : = 0.5850,		
V	λ_{44} : = 1.0000,	λ_{45} : = 0.3243,		
W	λ_{46} : = 1.0000,	λ_{47} : = 0.3243,		
O	λ_{48} : = 1.0000,	λ_{49} : = 0.2928,		
J	λ_{50} : = 0.6757,	λ_{51} : = 0.2191,		
K	λ_{52} : = 0.7072,	λ_{53} : = 0.2071,		
M	λ_{54} : = 1.0000,	λ_{55} : = 0.0510,	λ_{56} : = 0.0036,	λ57: = 0.0712,
	λ_{58} : = 0.0036,	λ_{59} : = 0.0003,		
X	λ_{60} : = 0.9288,	λ_{61} : = 0.1968,		
Z	λ_{62} : = 0.4211,	λ_{63} : = 0.5279,	λ_{64} : = 0.1119,	λ_{65} : = 0.0215,
	λ_{66} : = 0.0015,	λ_{67} : = 0.0300,	λ_{68} : = 0.0376,	λ_{69} : = 0.0080,
	λ_{70} : = 0.0015,	λ_{71} : = 0.0001,		
U	λ_{72} : = 0.7881,	λ_{73} : = 0.1670.		

Note that for $i \in \{74, \ldots, 105\}$, $\lambda_i := 1$ holds.

$$\mathrm{size}(\mathcal{I}, n) := \begin{cases} \mu_1 \cdot \mathrm{size}(\mathcal{E}, n) + \mu_2 \cdot \mathrm{size}(\mathcal{S}, n) + \mu_3 \cdot \mathrm{size}(\mathcal{A}, n) & \mathcal{I} = \mathcal{S}', \\ \mathrm{size}_E(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{E}_i \mid 1 \leq i \leq 4\} \text{ or } \mathcal{I} = \mathcal{E}, \\ \mathrm{size}_S(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{S}_i \mid 1 \leq i \leq 5\} \text{ or } \mathcal{I} = \mathcal{S}, \\ \mathrm{size}_C(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{C}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{C}, \\ \mathrm{size}_A(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{A}_i \mid 1 \leq i \leq 7\} \text{ or } \mathcal{I} = \mathcal{A}, \\ \mathrm{size}_G(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{G}_i \mid 1 \leq i \leq 4\} \text{ or } \mathcal{I} = \mathcal{G}, \\ \mathrm{size}_D(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{D}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{D}, \\ \mathrm{size}_B(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{B}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{B}, \\ \mathrm{size}_F(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{F}_i \mid 1 \leq i \leq 4\} \text{ or } \mathcal{I} = \mathcal{F}, \\ \mathrm{size}_H(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{H}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{H}, \\ \mathrm{size}_P(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{P}_i \mid 1 \leq i \leq 4\} \text{ or } \mathcal{I} = \mathcal{P}, \\ \mathrm{size}_Q(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{Q}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{Q}, \\ \mathrm{size}_R(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{R}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{R}, \\ \mathrm{size}_V(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{V}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{V}, \\ \mathrm{size}_W(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{W}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{W}, \\ \mathrm{size}_O(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{O}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{O}, \\ \mathrm{size}_J(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{J}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{J}, \\ \mathrm{size}_K(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{K}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{K}, \\ \mathrm{size}_M(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{M}_i \mid 1 \leq i \leq 6\} \text{ or } \mathcal{I} = \mathcal{M}, \\ \mathrm{size}_X(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{X}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{X}, \\ \mathrm{size}_Z(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{Z}_i \mid 1 \leq i \leq 10\} \text{ or } \mathcal{I} = \mathcal{Z}, \\ \mathrm{size}_U(\mathcal{I}, n) & \mathcal{I} \in \{\mathcal{U}_i \mid 1 \leq i \leq 2\} \text{ or } \mathcal{I} = \mathcal{U}, \\ 0 & \text{else}, \end{cases}$$

Table 6 Integer weights μ_i, $1 \le i \le 73$, for the production rules of the grammar $\hat{\mathcal{G}}^*_{sto}$

Nonterminal *Nt*	Integer weights of Rules with Premise *Nt*	
S′	$\mu_1 := 10000$,	$\mu_2 := 212$,
	$\mu_3 := 3$,	
E	$\mu_4 := 9788$,	$\mu_5 := 134$,
	$\mu_6 := 944$,	$\mu_7 := 13$,
S	$\mu_8 := 8559$,	$\mu_9 := 1304$,
	$\mu_{10} := 126$,	$\mu_{11} := 181$,
	$\mu_{12} := 2$,	
C	$\mu_{13} := 9036$,	$\mu_{14} := 871$,
A	$\mu_{15} := 76300000$,	$\mu_{16} := 4020000$,
	$\mu_{17} := 1860000$,	$\mu_{18} := 3670000$,
	$\mu_{19} := 720000$,	$\mu_{20} := 8580000$,
	$\mu_{21} := 4840000$,	
G	$\mu_{22} := 3038$,	$\mu_{23} := 1884$,
	$\mu_{24} := 3081$,	$\mu_{25} := 1996$,
D	$\mu_{26} := 10000$,	$\mu_{27} := 3896$,
B	$\mu_{28} := 6104$,	$\mu_{29} := 2378$,
F	$\mu_{30} := 5750000$,	$\mu_{31} := 340900000000$,
	$\mu_{32} := 6016000000000000$,	$\mu_{33} := 1211000000000000$,
H	$\mu_{34} := 7987$,	$\mu_{35} := 1608$,
P	$\mu_{36} := 10850000$,	$\mu_{37} := 214400000000$,
	$\mu_{38} := 201100000000$,	$\mu_{39} := 4760000000000000$,
Q	$\mu_{40} := 1713000000000000$,	$\mu_{41} := 828700000000$,
R	$\mu_{42} := 415000000000$,	$\mu_{43} := 585000000000$,
V	$\mu_{44} := 10000$,	$\mu_{45} := 3243$,
W	$\mu_{46} := 10000$,	$\mu_{47} := 3243$,
O	$\mu_{48} := 10000$,	$\mu_{49} := 2928$,
J	$\mu_{50} := 6757$,	$\mu_{51} := 2191$,
K	$\mu_{52} := 7072$,	$\mu_{53} := 2071$,
M	$\mu_{54} := 10000$,	$\mu_{55} := 510$,
	$\mu_{56} := 36$,	$\mu_{57} := 712$,
	$\mu_{58} := 36$,	$\mu_{59} := 3$,
X	$\mu_{60} := 9288$,	$\mu_{61} := 1968$,
Z	$\mu_{62} := 4211$,	$\mu_{63} := 5279$,
	$\mu_{64} := 1119$,	$\mu_{65} := 215$,
	$\mu_{66} := 15$,	$\mu_{67} := 300$,
	$\mu_{68} := 376$,	$\mu_{69} := 80$,
	$\mu_{70} := 15$,	$\mu_{71} := 1$,
U	$\mu_{72} := 7881$,	$\mu_{73} := 1670$.

Note that for $i \in \{74, \ldots, 105\}$, $\mu_i := 1$ holds.

where

$$\mathtt{size}_E(\mathcal{I}, n) := \begin{cases} \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{S}, j) \cdot \mathtt{size}(\mathcal{C}, n-j) & \mathcal{I} = \mathcal{E}_1, \\ \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{A}, j) \cdot \mathtt{size}(\mathcal{C}, n-j) & \mathcal{I} = \mathcal{E}_2, \\ \mathtt{size}(\mathcal{S}, n-1) & \mathcal{I} = \mathcal{E}_3, \\ \mathtt{size}(\mathcal{A}, n-1) & \mathcal{I} = \mathcal{E}_4, \\ \mu_4 \cdot \mathtt{size}(\mathcal{E}_1, n) + \mu_5 \cdot \mathtt{size}(\mathcal{E}_2, n) + \mu_6 \cdot \mathtt{size}(\mathcal{E}_3, n) + \mu_7 \cdot \mathtt{size}(\mathcal{E}_4, n) & \mathcal{I} = \mathcal{E}, \\ 0 & \text{else}, \end{cases}$$

$$\mathtt{size}_S(\mathcal{I},n) := \begin{cases} \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{E},j)\cdot\mathtt{size}(\mathcal{A},n-j) & \mathcal{I}=\mathcal{S}_1,\\ \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{C},j)\cdot\mathtt{size}(\mathcal{A},n-j) & \mathcal{I}=\mathcal{S}_2,\\ \mathtt{size}(\mathcal{A},n-1) & \mathcal{I}=\mathcal{S}_3,\\ \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{S},j)\cdot\mathtt{size}(\mathcal{A},n-j) & \mathcal{I}=\mathcal{S}_4,\\ \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{A},j)\cdot\mathtt{size}(\mathcal{A},n-j) & \mathcal{I}=\mathcal{S}_5,\\ \mu_8\cdot\mathtt{size}(\mathcal{S}_1,n)+\mu_9\cdot\mathtt{size}(\mathcal{S}_2,n)+\mu_{10}\cdot\mathtt{size}(\mathcal{S}_3,n) & \\ +\mu_{11}\cdot\mathtt{size}(\mathcal{S}_4,n)+\mu_{12}\cdot\mathtt{size}(\mathcal{S}_5,n) & \mathcal{I}=\mathcal{S},\\ 0 & \text{else,} \end{cases}$$

$$\mathrm{size}_C(\mathcal{I},n) := \begin{cases} \mathrm{size}(\mathcal{C},n-1) & \mathcal{I}=\mathcal{C}_1,\\ 1 & \mathcal{I}=\mathcal{C}_2,\ \text{and } n=2,\\ \mu_{13}\cdot\mathrm{size}(\mathcal{C}_1,n)+\mu_{14}\cdot\mathrm{size}(\mathcal{C}_2,n) & \mathcal{I}=\mathcal{C},\\ 0 & \text{else,} \end{cases}$$

$$\mathtt{size}_A(\mathcal{I},n) := \begin{cases} \mathtt{size}(\mathcal{A},n-2) & \mathcal{I}=\mathcal{A}_1,\\ \mathtt{size}(\mathcal{M},n-2) & \mathcal{I}=\mathcal{A}_2,\\ \mathtt{size}(\mathcal{P},n-2) & \mathcal{I}=\mathcal{A}_3,\\ \mathtt{size}(\mathcal{Q},n-2) & \mathcal{I}=\mathcal{A}_4,\\ \mathtt{size}(\mathcal{R},n-2) & \mathcal{I}=\mathcal{A}_5,\\ \mathtt{size}(\mathcal{F},n-2) & \mathcal{I}=\mathcal{A}_6,\\ \mathtt{size}(\mathcal{G},n-2) & \mathcal{I}=\mathcal{A}_7,\\ \mu_{15}\cdot\mathtt{size}(\mathcal{A}_1,n)+\mu_{16}\cdot\mathtt{size}(\mathcal{A}_2,n)+\mu_{17}\cdot\mathtt{size}(\mathcal{A}_3,n)+\mu_{18}\cdot\mathtt{size}(\mathcal{A}_4,n) & \\ +\mu_{19}\cdot\mathtt{size}(\mathcal{A}_5,n)+\mu_{20}\cdot\mathtt{size}(\mathcal{A}_6,n)+\mu_{21}\cdot\mathtt{size}(\mathcal{A}_7,n) & \mathcal{I}=\mathcal{A},\\ 0 & \text{else,} \end{cases}$$

$$\mathtt{size}_G(\mathcal{I},n) := \begin{cases} \mathtt{size}(\mathcal{A},n-1) & \mathcal{I}=\mathcal{G}_1,\\ \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{A},j)\cdot\mathtt{size}(\mathcal{D},n-j) & \mathcal{I}=\mathcal{G}_2,\\ \mathtt{size}(\mathcal{A},n-1) & \mathcal{I}=\mathcal{G}_3,\\ \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{D},j)\cdot\mathtt{size}(\mathcal{A},n-j) & \mathcal{I}=\mathcal{G}_4,\\ \mu_{22}\cdot\mathtt{size}(\mathcal{G}_1,n)+\mu_{23}\cdot\mathtt{size}(\mathcal{G}_2,n)+\mu_{24}\cdot\mathtt{size}(\mathcal{G}_3,n)+\mu_{25}\cdot\mathtt{size}(\mathcal{G}_4,n) & \mathcal{I}=\mathcal{G},\\ 0 & \text{else,} \end{cases}$$

$$\mathrm{size}_D(\mathcal{I},n) := \begin{cases} \mathrm{size}(\mathcal{B},n-1) & \mathcal{I}=\mathcal{D}_1,\\ 1 & \mathcal{I}=\mathcal{D}_2 \text{ and } n=2,\\ \mu_{26}\cdot\mathrm{size}(\mathcal{D}_{1,n})+\mu_{27}\cdot\mathrm{size}(\mathcal{D}_{2,n}) & \mathcal{I}=\mathcal{D},\\ 0 & \text{else,} \end{cases}$$

$$\mathrm{size}_B(\mathcal{I},n) := \begin{cases} \mathrm{size}(\mathcal{B},n-1) & \mathcal{I}=\mathcal{B}_1,\\ 1 & \mathcal{I}=\mathcal{B}_2 \text{ and } n=2,\\ \mu_{28}\cdot\mathrm{size}(\mathcal{B}_{1,n})+\mu_{29}\cdot\mathrm{size}(\mathcal{B}_{2,n}) & \mathcal{I}=\mathcal{B},\\ 0 & \text{else,} \end{cases}$$

$$\mathrm{size}_F(\mathcal{I},n) := \begin{cases} 1 & \mathcal{I}=\mathcal{F}_1 \text{ and } n=3,\\ 1 & \mathcal{I}=\mathcal{F}_2 \text{ and } n=4,\\ \mathrm{size}(\mathcal{H},n-4) & \mathcal{I}=\mathcal{F}_3,\\ 1 & \mathcal{I}=\mathcal{F}_4 \text{ and } n=5,\\ \mu_{30}\cdot\mathrm{size}(\mathcal{F}_1,n)+\mu_{31}\cdot\mathrm{size}(\mathcal{F}_2,n) & \\ +\mu_{32}\cdot\mathrm{size}(\mathcal{F}_3,n)+\mu_{33}\cdot\mathrm{size}(\mathcal{F}_4,n) & \mathcal{I}=\mathcal{F},\\ 0 & \text{else,} \end{cases}$$

$$\text{size}_H(\mathcal{I}, n) := \begin{cases} \text{size}(\mathcal{H}, n-1) & \mathcal{I} = \mathcal{H}_1, \\ 1 & \mathcal{I} = \mathcal{H}_2, \text{ and } n = 2 \\ \mu_{34} \cdot \text{size}(\mathcal{H}_1, n) + \mu_{35} \cdot \text{size}(\mathcal{H}_2, n) & \mathcal{I} = \mathcal{H}, \\ 0 & \text{else}, \end{cases}$$

$$\text{size}_P(\mathcal{I}, n) := \begin{cases} \text{size}(\mathcal{A}, n-2) & \mathcal{I} = \mathcal{P}_1, \\ \text{size}(\mathcal{A}, n-3) & \mathcal{I} = \mathcal{P}_2, \\ \text{size}(\mathcal{A}, n-3) & \mathcal{I} = \mathcal{P}_3, \\ \text{size}(\mathcal{A}, n-4) & \mathcal{I} = \mathcal{P}_4, \\ \mu_{36} \cdot \text{size}(\mathcal{P}_1, n) + \mu_{37} \cdot \text{size}(\mathcal{P}_2, n) + \mu_{38} \cdot \text{size}(\mathcal{P}_3, n) + \mu_{39} \cdot \text{size}(\mathcal{P}_4, n) & \mathcal{I} = \mathcal{P}, \\ 0 & \text{else}, \end{cases}$$

$$\text{size}_Q(\mathcal{I}, n) := \begin{cases} \text{size}(\mathcal{O}, n-4) & \mathcal{I} = \mathcal{Q}_1, \\ \text{size}(\mathcal{V}, n-3) & \mathcal{I} = \mathcal{Q}_2, \\ \mu_{40} \cdot \text{size}(\mathcal{Q}_{1,n}) + \mu_{41} \cdot \text{size}(\mathcal{Q}_{2,n}) & \mathcal{I} = \mathcal{Q}, \\ 0 & \text{else}, \end{cases}$$

$$\text{size}_R(\mathcal{I}, n) := \begin{cases} \text{size}(\mathcal{O}, n-3) & \mathcal{I} = \mathcal{R}_1, \\ \text{size}(\mathcal{W}, n-3) & \mathcal{I} = \mathcal{R}_2, \\ \mu_{42} \cdot \text{size}(\mathcal{R}_{1,n}) + \mu_{43} \cdot \text{size}(\mathcal{R}_{2,n}) & \mathcal{I} = \mathcal{R}, \\ 0 & \text{else}, \end{cases}$$

$$\text{size}_V(\mathcal{I}, n) := \begin{cases} \sum_{j=1}^{n-1} \text{size}(\mathcal{J}, j) \cdot \text{size}(\mathcal{O}, n-j) & \mathcal{I} = \mathcal{V} \\ \text{size}(\mathcal{O}, n-1) & \mathcal{I} = \mathcal{V}_2, \\ \mu_{44} \cdot \text{size}(\mathcal{V}_1, n) + \mu_{45} \cdot \text{size}(\mathcal{V}_2, n) & \mathcal{I} = \mathcal{V}, \\ 0 & \text{else}, \end{cases}$$

$$\text{size}_W(\mathcal{I}, n) := \begin{cases} \sum_{j=1}^{n-1} \text{size}(\mathcal{J}, j) \cdot \text{size}(\mathcal{A}, n-j) & \mathcal{I} = \mathcal{W}_1 \\ \text{size}(\mathcal{A}, n-1) & \mathcal{I} = \mathcal{W}_2, \\ \mu_{46} \cdot \text{size}(\mathcal{W}_1, n) + \mu_{47} \cdot \text{size}(\mathcal{W}_2, n) & \mathcal{I} = \mathcal{W}, \\ 0 & \text{else}, \end{cases}$$

$$\text{size}_O(\mathcal{I}, n) := \begin{cases} \sum_{j=1}^{n-1} \text{size}(\mathcal{A}, j) \cdot \text{size}(\mathcal{K}, n-j) & \mathcal{I} = \mathcal{O}_1 \\ \text{size}(\mathcal{A}, n-1) & \mathcal{I} = \mathcal{O}_2, \\ \mu_{48} \cdot \text{size}(\mathcal{O}_1, n) + \mu_{49} \cdot \text{size}(\mathcal{O}_2, n) & \mathcal{I} = \mathcal{O}, \\ 0 & \text{else}, \end{cases}$$

$$\text{size}_J(\mathcal{I}, n) := \begin{cases} \text{size}(\mathcal{J}, n-1) & \mathcal{I} = \mathcal{J}_1, \\ 1 & \mathcal{I} = \mathcal{J}_2 \text{ and } n = 2, \\ \mu_{50} \cdot \text{size}(\mathcal{J}_1, n) + \mu_{51} \cdot \text{size}(\mathcal{J}_2, n) & \mathcal{I} = \mathcal{J} \\ 0 & \text{else}, \end{cases}$$

$$\text{size}_K(\mathcal{I}, n) := \begin{cases} \text{size}(\mathcal{K}, n-1) & \mathcal{I} = \mathcal{K}_1, \\ 1 & \mathcal{I} = \mathcal{J}_2 \text{ and } n = 2, \\ \mu_{52} \cdot \text{size}(\mathcal{K}_1, n) + \mu_{53} \cdot \text{size}(\mathcal{K}_2, n) & \mathcal{I} = \mathcal{K}, \\ 0 & \text{else}, \end{cases}$$

$$\mathtt{size}_M(\mathcal{I},n) := \begin{cases} \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{X},j)\cdot\mathtt{size}(\mathcal{Z},n-j) & \mathcal{I}=\mathcal{M}_1,\\ \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{X},j)\cdot\mathtt{size}(\mathcal{X},n-j) & \mathcal{I}=\mathcal{M}_2,\\ \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{X},j)\cdot\mathtt{size}(\mathcal{A},n-j) & \mathcal{I}=\mathcal{M}_3,\\ \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{A},j)\cdot\mathtt{size}(\mathcal{Z},n-j) & \mathcal{I}=\mathcal{M}_4,\\ \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{A},j)\cdot\mathtt{size}(\mathcal{X},n-j) & \mathcal{I}=\mathcal{M}_5,\\ \sum_{j=1}^{n-1} \mathtt{size}(\mathcal{A},j)\cdot\mathtt{size}(\mathcal{A},n-j) & \mathcal{I}=\mathcal{M}_6,\\ \mu_{54}\cdot\mathtt{size}(\mathcal{M}_1,n)+\mu_{55}\cdot\mathtt{size}(\mathcal{M}_2,n)+\mu_{56}\cdot\mathtt{size}(\mathcal{M}_3,n) & \\ +\mu_{57}\cdot\mathtt{size}(\mathcal{M}_4,n)+\mu_{58}\cdot\mathtt{size}(\mathcal{M}_5,n)+\mu_{59}\cdot\mathtt{size}(\mathcal{M}_6,n) & \mathcal{I}=\mathcal{M},\\ 0 & \text{else}, \end{cases}$$

$$\mathrm{size}_X(\mathcal{I},n) := \begin{cases} \sum_{j=1}^{n-1} \mathrm{size}(\mathcal{U},j)\cdot\mathrm{size}(\mathcal{A},n-j) & \mathcal{I}=\mathcal{X}_1\\ \mathrm{size}(\mathcal{A},n-1) & \mathcal{I}=\mathcal{X}_2,\\ \mu_{60}\cdot\mathrm{size}(\mathcal{X}_1,n)+\mu_{61}\cdot\mathrm{size}(\mathcal{X}_2,n) & \mathcal{I}=\mathcal{X},\\ 0 & \text{else}, \end{cases}$$

$$\mathrm{size}_z(\mathcal{I},n) := \begin{cases} \sum_{j=1}^{n-1} \mathrm{size}(\mathcal{X},j)\cdot\mathrm{size}(\mathcal{Z},n-j) & \mathcal{I}=\mathcal{Z}_1,\\ \sum_{j=1}^{n-1} \mathrm{size}(\mathcal{X},j)\cdot\mathrm{size}(\mathcal{U},n-j) & \mathcal{I}=\mathcal{Z}_2,\\ \mathrm{size}(\mathcal{X},n-1) & \mathcal{I}=\mathcal{Z}_3,\\ \sum_{j=1}^{n-1} \mathrm{size}(\mathcal{X},j)\cdot\mathrm{size}(\mathcal{X},n-j) & \mathcal{I}=\mathcal{Z}_4,\\ \sum_{j=1}^{n-1} \mathrm{size}(\mathcal{X},j)\cdot\mathrm{size}(\mathcal{A},n-j) & \mathcal{I}=\mathcal{Z}_5,\\ \sum_{j=1}^{n-1} \mathrm{size}(\mathcal{A},j)\cdot\mathrm{size}(\mathcal{Z},n-j) & \mathcal{I}=\mathcal{Z}_6,\\ \sum_{j=1}^{n-1} \mathrm{size}(\mathcal{A},j)\cdot\mathrm{size}(\mathcal{U},n-j) & \mathcal{I}=\mathcal{Z}_7,\\ \mathrm{size}(\mathcal{A},n-1) & \mathcal{I}=\mathcal{Z}_8,\\ \sum_{j=1}^{n-1} \mathrm{size}(\mathcal{A},j)\cdot\mathrm{size}(\mathcal{A},n-j) & \mathcal{I}=\mathcal{Z}_9,\\ \sum_{j=1}^{n-1} \mathrm{size}(\mathcal{A},j)\cdot\mathrm{size}(\mathcal{A},n-j) & \mathcal{I}=\mathcal{Z}_{10},\\ \mu_{62}\cdot\mathrm{size}(\mathcal{Z}_1,n)+\mu_{63}\cdot\mathrm{size}(\mathcal{Z}_2,n)+\mu_{64}\cdot\mathrm{size}(\mathcal{Z}_3,n)+\mu_{65}\cdot\mathrm{size}(\mathcal{Z}_4,n) & \\ +\mu_{66}\cdot\mathrm{size}(\mathcal{Z}_5,n)+\mu_{67}\cdot\mathrm{size}(\mathcal{Z}_6,n)+\mu_{68}\cdot\mathrm{size}(\mathcal{Z}_7,n)+\mu_{69}\cdot\mathrm{size}(\mathcal{Z}_8,n) & \\ +\mu_{70}\cdot\mathrm{size}(\mathcal{Z}_9,n)+\mu_{71}\cdot\mathrm{size}(\mathcal{Z}_{10},n) & \mathcal{I}=\mathcal{Z},\\ 0 & \text{else}, \end{cases}$$

$$\mathrm{size}_U(\mathcal{I},n) := \begin{cases} \mathrm{size}(\mathcal{U},n-1) & \mathcal{I}=\mathcal{U}_1,\\ 1 & \mathcal{I}=\mathcal{U}_2, \quad \text{and} \quad n=2\\ \mu_{72}\cdot\mathrm{size}(\mathcal{U}_1,n)+\mu_{73}\cdot\mathrm{size}(\mathcal{U}_2,n) & \mathcal{I}=\mathcal{U},\\ 0 & \text{else}. \end{cases}$$

From those recurrences, the desired algorithm can easily be constructed. As the complete presentation of this algorithm would be too comprehensive, we decided to omit it and instead refer to Algorithms 1 to 4 and 6 given in [20], since for the construction of our unranking algorithm, we had to use exactly these Algorithms as subroutines.

Acknowledgements
AS thanks Carl-Zeiss-Stiftung for supporting her research. All authors wish to thank an anonymous reviewer for careful reading and helpful remarks and suggestions made for a previous version of this article.

Authors' contributions
MEN and FW have invented the general framework for the non-uniform random generation. AS and MEN designed the SCFG for RNA secondary structures; MEN proved its unambiguity. AS developed and implemented the algorithms for generating random RNA secondary structures. AS performed all experiments and evaluated the quality of our algorithms. MEN supervised the work and development of ideas. AS drafted the manuscript; a revision and its final version have been prepared by MEN. All authors have read and approved the final manuscript.

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

Received: 20 April 2011 Accepted: 12 October 2011 Published: 12 October 2011

References

1. Flajolet P, Fusy E, Pivoteau C: **Boltzmann Sampling of Unlabelled Structures.** *Proceedings of ANALCO'07 (Analytic Combinatorics and Algorithms) Conference* SIAM Press; 2007, 201-211.
2. Fitch WM: **Random sequences.** *Journal of Molecular Biology* 1983, **163**:171-176.
3. Altschul SF, Erickson BW: **Significance of nucleotide sequence alignments: a method for random sequence permutation that preserves dinucleotide and codon usage.** *Mol Biol Evol* 1985, **2(6)**:256-538.
4. Denise A, Ponty Y, Termier M: **Random Generation of structured genomic sequences.** *Proceedings of RECOMB 2003* 2003, 3, (poster).
5. Waterman MS: **Secondary Structure of Single-Stranded Nucleic Acids.** *Advances in Mathematics Supplementary Studies* 1978, **1**:167-212.
6. Ding Y, Lawrence CE: **A statistical sampling algorithm for RNA secondary structure prediction.** *Nucleic Acids Research* 2003, **31(24)**:7280-7301.
7. Ponty Y: **Efficient sampling of RNA secondary structures from the Boltzmann ensemble of low-energy: the boustrophedon method.** *Journal of Mathematical Biology* 2008, **56**:107-127.
8. Allali J, d'Aubenton Carafa Y, Chauve C, Denise A, Drevet C, Ferraro P, Gautheret D, Herrbach C, Leclerc F, de Monte A, Ouangraoua A, Sagot MF, Saule C, Termier M, Thermes C, Touzet H: **Benchmarking RNA secondary structures comparison algorithms.** *Actes des Journées Ouvertes de Biologie, Informatique et Mathématiques - JOBIM'08* 2008, 67-68.
9. Wuchty S, Fontana W, Hofacker I, Schuster P: **Complete Suboptimal Folding of RNA and the Stability of Secondary Structures.** *Biopolymers* 1999, **49**:145-165.
10. Zuker M: **On Finding All Suboptimal Foldings of an RNA Molecule.** *Science* 1989, **244**:48-52.
11. Zuker M: **Mfold web server for nucleic acid folding and hybridization prediction.** *Nucleic Acids Res* 2003, **31(13)**:3406-3415.
12. Hofacker IL: **The Vienna RNA secondary structure server.** *Nucleic Acids Research* 2003, **31(13)**:3429-3431.
13. Dowell RD, Eddy SR: **Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction.** *BMC Bioinformatics* 2004, **5**:71.
14. Knudsen B, Hein J: **RNA secondary structure prediction using stochastic context-free grammars and evolutionary history.** *Bioinformatics* 1999, **15(6)**:446-454.
15. Knudsen B, Hein J: **Pfold: RNA secondary structure prediction using stochastic context-free grammars.** *Nucleic Acids Research* 2003, **31(13)**:3423-3428.
16. Pedersen J, Meyer I, Forsberg R, Simmonds P, Hein J: **A comparative method for finding and folding RNA secondary structures in protein-coding regions.** *Nucleic Acids Reserach* 2004, **32**:4925-4936.
17. Pedersen JS, Forsberg R, Meyer IM, Hein J: **An Evolutionary Model for Protein-Coding Regions with Conserved RNA Structure.** *Molecular Biology and Evolution* 2004, **21**:1913-1922.
18. Wiebe NJP, Meyer IM: **¡sc¿Transat¡/sc¿A Method for Detecting the Conserved Helices of Functional RNA Structures, Including Transient, Pseudo-Knotted and Alternative Structures.** *PLoS Comput Biol* 2010, **6(6)**:e1000823.
19. Gesell T, von Haeseler A: **In silico sequence evolution with site-specific interactions along phylogenetic trees.** *Bioinformatics* 2006, **22**:716-722.
20. Weinberg F, Nebel ME: **Non Uniform Generation of Combinatorial Objects.** Tech. rep., Technische Universität Kaiserslautern; 2010.
21. Nebel ME, Scheid A: **Analysis of the Free Energy in a Stochastic RNA Secondary Structure Model.** *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2010.
22. Xia T, SantaLucia J Jr, Burkard ME, Kierzek R, Schroeder SJ, Jiao X, Cox C, Turner DH: **Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs.** *Biochemistry* 1998, **37**:14719-14735.
23. Mathews DH, Sabina J, Zuker M, Turner DH: **Expanded Sequence Dependence of Thermodynamic Parameters Improves Prediction of RNA Secondary Structure.** *J Mol Biol* 1999, **288**:911-940.
24. Nijenhuis A, Wilf HS: *Combinatorial Algorithms* , 2 1978, Academic Press.
25. Flajolet P, Zimmermann P, Van Cutsem B: **A Calculus for the Random Generation of Combinatorial Structures.** *Theoretical Computer Science* 1994, **132(2)**:1-35.
26. Duchon P, Flajolet P, Louchard G, Schaeffer G: **Boltzmann Samplers for the Random Generation of Combinatorial Structures.** *Combinatorics, Probability, and Computing, Volume 13* 2004, 577-625, [Special issue on Analysis of Algorithms].
27. Flajolet P, Sedgewick R: *Analytic Combinatorics* Cambridge University Press; 2009.
28. Harrison MA: *Introduction to Formal Language Theory* Addison-Wesley; 1978.
29. Stein PR, Waterman MS: **On some new sequences generalizing the Catalan and Motzkin numbers.** *Discrete Mathematics* 1978, **26**:216-272.
30. Viennot G, Chaumont MVD: **Enumeration of RNA Secondary Structures by Complexity.** *Mathematics in medicine and biology, Lecture Notes in Biomathematics* 1985, **57**:360-365.
31. Nebel ME: **Combinatorial Properties of RNA Secondary Structures.** *Journal of Computational Biology* 2002, **9(3)**:541-574.
32. Hofacker IL, Schuster P, Stadler PF: **Combinatorics of RNA secondary structures.** *Discrete Applied Mathematics* 1998, **88**:207-237.
33. Nebel ME: **Investigation of the Bernoulli-Model of RNA Secondary Structures.** *Bulletin of Mathematical Biology* 2004, **66**:925-964.
34. Zuker M, Sankoff D: **RNA Secondary Structures and their Prediction.** *Bull Mathematical Biology* 1984, **46**:591-621.
35. Nebel ME: **On a statistical filter for RNA secondary structures.** Tech. rep., Frankfurter Informatik-Berichte; 2002.
36. Nebel ME: **Identifying Good Predictions of RNA Secondary Structure.** *Proceedings of the Pacific Symposium on Biocomputing* 2004, 423-434.
37. Molinero X: **Ordered Generation of Classes of Combinatorial Structures.** *PhD thesis* Universitat Politècnica de Catalunya; 2005.
38. Fu KS, Huang T: **Stochastic Grammars and Languages.** *International Journal of Computer and Information Sciences* 1972, **1(2)**:135-170.

39. Huang T, Fu KS: **On Stochastic Context-Free Languages.** *Information Sciences* 1971, **3**:201-224.
40. Sakakibara Y, Brown M, Hughey R, Mian IS, Sjölander K, Underwood RC, Haussler D: **Stochastic context-free grammars for tRNA modeling.** *Nucleic Acids Research* 1994, **22**:5112-5120.
41. Liebehenschel J: **Ranking and unranking of lexicographically ordered words: an average-case analysis.** *J Autom Lang Comb* 1998, **2(4)**:227-268.
42. Weinberg F, Nebel ME: **Extending Stochastic Context-Free Grammars for an Application in Bioin-formatics.** *4th International Conference on Language and Automata Theory and Applications (LATA2010)* 2010.
43. Nawrocki EP, Eddy SR: **Query-Dependent Banding (QDB) for Faster RNA Similarity Searches.** *PLoS Comput Biol* 2007, **3**:e56.
44. Martínez C, Molinero X: **A generic approach for the unranking of labeled combinatorial classes. Random Struct.** *Algorithms* 2001, **19(3-4)**:472-497.
45. Wuyts J, Rijk PD, de Peer YV, Winkelmans T, Wachter RD: **The European Large Subunit Ribosomal RNA Database.** *Nucleic Acids Research* 2001, **29**:175-177.
46. Wuyts J, de Peer YV, Winkelmans T, Wachter RD: **The European Database on Small Subunit Ribosomal RNA.** *Nucleic Acids Research* 2002, **30**:183-185.
47. Salomaa A, Soittola M: *Automata-theoretic aspects of formal power series* Springer; 1978.
48. Mann H, Whitney D: **On a test of whether one of two random variables is stochastically larger than the other.** *Annals of Mathematical Statistics* 1947, **18**:50-60.
49. Wilcoxon F: **Individual Comparisons by Ranking Methods.** *Biometrics Bulletin* 1945, **1**:80-83.

doi:10.1186/1748-7188-6-24
Cite this article as: Nebel *et al.*: Random generation of RNA secondary structures according to native distributions. *Algorithms for Molecular Biology* 2011 **6**:24.