International Journal of Foundations of Computer Science

Counting, generating, analyzing and sampling tree alignments

Cedric Chauve

Department of Mathematics, Simon Fraser University
8888 University Drive, Burnaby, B.C. V5A 1S6, Canada
cedric.chauve@sfu.ca

Julien Courtiel

Laboratoire d'Informatique de Paris Nord, Université Paris 13
99, avenue Jean-Baptiste Clément 93430 Villetaneuse, France
courtiel@lipn.univ-paris13.fr

Yann Ponty

LIX CNRS UMR 7161, Ecole Polytechnique, Université Paris-Saclay
AMIBio team, Inria Saclay, Université Paris-Saclay
Bat. Alan Turing, 1 rue d'Estienne d'Orves Palaiseau, 91120, France
yann.ponty@lix.polytechnique.fr

Received (Day Month Year)

Accepted (Day Month Year)

Communicated by (xxxxxxxxxx)

Pairwise ordered tree alignment are combinatorial objects that appear in important applications, such as RNA secondary structure comparison. However, the usual representation of tree alignments as supertrees is ambiguous, i.e. two distinct supertrees may induce identical sets of matches between identical pairs of trees. This ambiguity is uninformative, and detrimental to any probabilistic analysis.

In this work, we consider tree alignments up to equivalence. Our first result is a precise asymptotic enumeration of tree alignments, obtained from a context-free grammar by mean of basic analytic combinatorics. Our second result focuses on alignments between two given ordered trees S and T. By refining our grammar to align specific trees, we obtain a decomposition scheme for the space of alignments, and use it to design an efficient dynamic programming algorithm for sampling alignments under the Gibbs-Boltzmann probability distribution. This generalizes existing tree alignment algorithms, and opens the door for a probabilistic analysis of the space of suboptimal alignments.

Keywords: Tree alignments; Analytic combinatorics; Gibbs/Bolzmann sampling; Average-case complexity analysis.

1. Introduction

Tree alignments are the natural analog of sequence alignments, and have been introduced by Jiang, Wang and Zhang [16] to model and quantify the similarity between two (ordered[a]) trees. Initially proposed as an alternative to tree-edit distance, the tree alignment model has proven more robust, allowing for the inclusion of complex local operations [3], and for being generalized to multiple input trees [15]. Consequently, tree alignment has been used in a wide array of applicative contexts, especially RNA Bioinformatics [14], where RNA secondary structures alignments can be encoded by tree alignments. The minimal cost tree alignment between two trees of size n_1 and n_2, under classic insertion/deletion/(mis)-match operations, can be computed using dynamic programming (DP). The current best algorithms have a worst-case time and space complexity respectively in $\mathcal{O}(n_1 n_2 (n_1+n_2)^2)$ and $\mathcal{O}(n_1 n_2 (n_1+n_2))$ [16], and an average-case time and space complexity (on uniformly drawn instances) in $\mathcal{O}(n_1 n_2)$ [13].

In the context of sequence alignments, the enumeration of alignments has been the object of much interest in Computational Biology [6, 21, 2]. Alignments between two sequences over an alphabet Σ can be encoded as sequences over an extended alphabet Σ_a, representing insertions, deletions and (mis)matches (*e.g.* $\Sigma = \{a, b\}$, $\Sigma_a = \{(a,-),(-,b),(a,b),(a,a),(b,a),(b,b)\}$). Many sequences over Σ_a are equivalent if one considers only (mis)matches of the alignments, *i.e.* they align sequence of same lengths and induce the same sets of matched positions (*e.g.* $(a,-),(-,b)$ and $(-,b),(a,-)$). It is a natural problem to enumerate distinct sequence alignments for two sequences of cumulated length n [23] . Beyond purely theoretical considerations, the decompositions introduced for enumerating distinct sequence alignments were adapted into DP algorithms, *e.g.* for probabilistic alignment based on expectation maximization [5], or to compute Gibbs-Boltzmann measures of reliability [22].

In the present work, we consider similar questions on *tree alignments*. We are first interested in counting distinct tree alignments, *i.e.* enumerating, up to equivalence, ordered trees whose vertices are labeled in Σ_a (called *supertrees* from now). For trees, the notion of equivalence of alignments generalizes that of sequence alignments, *i.e.* two alignments are *equivalent* when they align the same pairs of trees, and induce the same sets of (mis)matched positions. Unfortunately, contrasting with the case of sequence alignments, existing DP algorithms for computing an optimal tree alignment [16, 3, 20] cannot be easily adapted into enumeration schemes for tree alignments up to equivalence. This additional difficulty is due to the existence of ambiguities of different nature.

Our main contribution is a grammar for (distinct) tree alignments, which provably generates a single representative for each equivalence class. We use the sym-

[a]In this work, unless explicitly specified, all trees will be rooted and ordered.

bolic method [11] to obtain the generating function of tree alignments, and asymptotic equivalents for various statistics of interest can easily be derived, such as the average number of alignments over trees of total size n. Moreover, while arguably complex, this grammar is context-free, and random generators can be immediately derived using the recursive method [24] or Boltzmann sampling [8]. Finally, and, perhaps more importantly from an applied point of view, the grammar can be transformed into an unambiguous and complete DP algorithm for aligning two input trees. The resulting algorithm has the same asymptotic worst-case and average-case complexities, up to reasonable constants, as the current best – ambiguous – algorithm [16, 3]. The main interest of such an algorithm is that it opens immediately the way to new applications for the tree alignment model, including a critical assessment of the reliability of optimal alignments, either obtained by counting co-optimal alignments, or by sampling suboptimal alignments according to a Gibbs-Boltzmann distribution (see [18] for an example of this approach for the RNA folding problem).

In Section 2 we introduce the main definitions about trees, supertrees and tree alignments. In Section 3, we provide a grammar that generates all tree alignments. In Section 4.1 we analyze this grammar from an enumerative point of view and give precise results on the number of alignments of fixed size. Finally, in Section 4.2 we show how to transform the tree alignments grammar into a dynamic programming algorithm to sample tree alignments between two specified trees.

2. Definitions

Trees and supertrees. Let Σ be an *alphabet*. A tree T on Σ is a rooted plane tree whose vertices are labeled by elements of Σ. We denote by V_T the set of vertices of T. We *remove a non-root vertex* v from a tree T by contracting the edge between v and its parent u, that keeps its label. Removing the root r of a tree consists in creating a forest composed of the subtrees rooted at the children of r. We denote the operation of removing a vertex v from T by $T - v$.

We denote by Σ_a the alphabet defined by $\Sigma_a = (\Sigma \cup \{-\})^2 - \{(-,-)\}$. An element $(x, y) \in \Sigma_a$ is an *insertion* (resp. *deletion*, *match*) if $y = -$ (resp. $x = -$, $(x, y) \in \Sigma^2$). A *supertree* A is a tree on Σ_a; a vertex of A is an insertion (resp. deletion, match) if its label is an insertion (resp. deletion, match). The size of a supertree A is the number of its insertions and deletions, plus twice the number of its matches. A *superforest* is an ordered sequence of supertrees.

Given a supertree A on Σ, we define two forests $\pi_1(A)$ and $\pi_2(A)$ as follows: $\pi_1(A)$ (resp. $\pi_2(A)$) is obtained by (1) iteratively removing all insertion (resp. deletions) of A, in an arbitrary order, and (2) replacing the label (x, y) of each remaining vertex by x (resp. y). We refer to Fig. 1 for an illustration. We extend the notations π_1 and π_2 on vertices: for a non-insertion (resp. non-deletion) vertex v of A, we denote by $\pi_1(v)$ (resp. $\pi_2(v)$) the corresponding vertex in $\pi_1(A)$ (resp. $\pi_2(A)$). A

vertex x of $\pi_1(A)$ such that $\pi_1^{-1}(x)$ is an insertion (resp. match) is said to be inserted (resp. matched) in A. Similarly, a vertex y of $\pi_2(A)$ such that $\pi_2^{-1}(y)$ is a deletion (resp. match) is said to be deleted (resp. matched) in A.

Tree alignments. As forests $\pi_1(A)$ and $\pi_2(A)$ are embedded into the supertree A, the latter implicitly defines an *alignment* between the forests $\pi_1(A)$ and $\pi_2(A)$, *i.e.* a set of correspondences between vertices of $\pi_1(A)$ and $\pi_2(A)$, that is consistent with the structure of both forests [16]. We refer to Fig. 1 for an illustration.

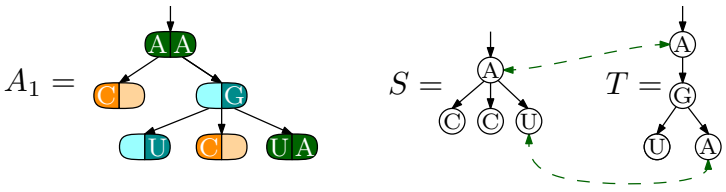

Figure 1. A supertree A_1 with alphabet $\Sigma = \{A, C, G, U\}$, and the associated trees $S = \pi_1(A_1)$ and $T = \pi_2(A_1)$. The alignment of S and T defined by A is composed of two pairs of matched (A, A) and (U, A), indicated by dashed arrows.

We now turn to the central notion of *equivalent alignments*, *i.e.* alignments of identical pairs of trees, that contain exactly the same set of matched vertices. Given a supertree A, representing an alignment between two trees $S = \pi_1(A)$ and $T = \pi_2(A)$, the *set of matches of* A is formed by the elements (x, y) of $V_S \times V_T$ such that $\pi_1^{-1}(x) = \pi_2^{-1}(y)$ (*i.e.* there exists a vertex v of A such that $\pi_1(v) = x$ and $\pi_2(v) = y$). Two supertrees A_1 and A_2 are *equivalent* if $\pi_1(A_1) = \pi_1(A_2)$, $\pi_2(A_1) = \pi_2(A_2)$, and the sets of matches of A_1 and A_2 are identical (see Fig. 2 for an illustration).

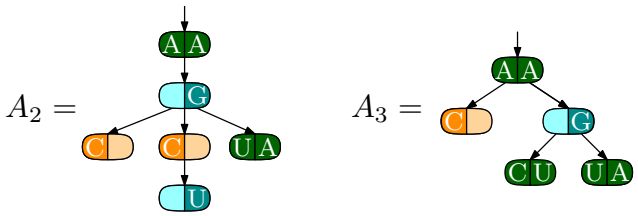

Figure 2. Two non-equivalent supertrees, representing two different tree alignments. However, the supertree A_1 from Fig. 1 and the supertree A_2 are equivalent.

A *tree alignment* is then defined as an equivalence class over supertrees with respect to the above-defined equivalence relation, for which $\pi_1(A)$ and $\pi_2(A)$ are trees. The notion of *forest alignment* is similarly defined when $\pi_1(A)$ and $\pi_2(A)$ are not restricted to trees. We extend the definitions of π_1 and π_2 for tree alignments.

Given a set $\mathcal{S}$ of tree (resp. forest) alignments, a set $\mathcal{T}$ of supertrees (resp. superforests) is said to be *representative of* $\mathcal{S}$ if it contains exactly one supertree (resp. superforest) for each alignment (*i.e.* equivalence classes of supertrees and forests) in $\mathcal{S}$. Tree alignments will now be the focus of our work.

Example 1. *The tree alignment A_1 from Fig. 1 and the tree alignment A_2 from Fig. 2 are equivalent: the induced trees are the same, as well as the set of matches. However, even if they have the same images under π_1 and π_2, the alignments A_1 and A_2 are not equivalent to the tree alignment A_3 depicted in Fig. 2: an extra vertex has been matched.*

3. A grammar for tree alignments

In this section, we describe a context-free grammar for a set $\mathcal{A}$ of supertrees that is representative of the set of all tree alignments. As mentioned in our introduction, this constitutes a delicate problem insofar as supertrees feature different levels of ambiguity. Consequently, the grammar we present is particularly involved.

We first define some basic operations on supertrees and superforests:

- The (ordered) concatenation of two (super)forests A and B is denoted by $A \circ B$. It creates a new superforest beginning by the supertrees of A, and ending by the supertrees of B.
- Given two disjoint sets $\mathcal{T}_1$ and $\mathcal{T}_2$ of supertrees or superforests, we denote by $\mathcal{T}_1 \oplus \mathcal{T}_2$ their (disjoint) union.
- For any superforest A and $a, b \in \Sigma$, $\mathrm{InsR}\,(A, a)$ (resp. $\mathrm{DelR}\,(A, b)$, $\mathrm{MatchR}\,(A, a, b)$) denotes the supertree whose root is the vertex $(a, -)$ (resp. $(-, b)$, (a, b)) and whose children are the supertrees in A, ordered with the same order that they have in A.
- We naturally extend these operators to a set $\mathcal{T}$ of supertrees or superforests:

$$\mathrm{InsR}\,(\mathcal{T}) = \bigoplus_{A \in \mathcal{T}, a \in \Sigma} \mathrm{InsR}\,(A, a)\,,$$

$$\mathrm{DelR}\,(\mathcal{T}) = \bigoplus_{A \in \mathcal{T}, a \in \Sigma} \mathrm{DelR}\,(A, a)\,,$$

$$\mathrm{MatchR}\,(\mathcal{T}) = \bigoplus_{A \in \mathcal{T}, (a,b) \in \Sigma^2} \mathrm{MatchR}\,(A, a, b)\,.$$

Our grammar is described in Fig. 3, and illustrated in Fig. 4. The terminal states of this grammar are given by the empty superforests.

The following theorem states that the previous grammar unambiguously generates all tree alignments as a representative set of supertrees.

$$\mathcal{A} = \mathcal{V}^{\varnothing} \oplus \mathrm{InsR}\left(\mathcal{F}_I \circ \mathcal{T}_D\right) \tag{1}$$

$$\mathcal{T}_I = \mathrm{InsR}\left(\mathcal{F}_I\right), \quad \mathcal{F}_I = \{\text{empty superforest}\} \oplus \mathrm{InsR}\left(\mathcal{F}_I\right) \circ \mathcal{F}_I \tag{2}$$

$$\mathcal{T}_D = \mathrm{InsR}\left(\mathcal{F}_D\right), \quad \mathcal{F}_D = \{\text{empty superforest}\} \oplus \mathrm{InsR}\left(\mathcal{F}_D\right) \circ \mathcal{F}_D \tag{3}$$

$$\mathcal{V}^{\varnothing} = \mathcal{V}^{\uparrow} \oplus \mathrm{InsR}\left(\mathcal{V}\mathcal{H}\right) \tag{4}$$

$$\mathcal{V}^{\uparrow} = \mathrm{MatchR}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\varnothing}\right) \oplus \mathrm{DelR}\left(\mathcal{F}_D \circ \mathcal{V}^{\uparrow} \circ \mathcal{F}_D\right) \tag{5}$$

$$\mathcal{V}\mathcal{H} = \mathcal{F}_I \circ \mathcal{V}\mathcal{H} \oplus \mathcal{V}^{\varnothing} \circ \mathcal{F}_I \oplus \mathrm{DelR}\left(\mathcal{H}_{\mathsf{I|D},\leftrightarrow,\varnothing}\right) \circ \mathcal{F}_I \tag{6}$$

For every ν, M, M' with $\nu \in \{\mathsf{I|D}, \mathsf{D}\}$ and $M, M' \in \{\varnothing, \leftrightarrow, \rightarrow\}$:

$$\mathcal{H}_{\nu,M,M'} = \bigoplus \begin{cases} \{\text{empty superforest}\} & \text{if } (M,M') = (\varnothing,\varnothing) \\ \mathcal{T}_I \circ \mathcal{H}_{\nu,M,M'} & \text{if } \nu \neq \mathsf{D} \text{ and if } M \neq \leftrightarrow \\ \mathcal{T}_D \circ \mathcal{H}_{\mathsf{D},M,M'} & \text{if } M' \neq \leftrightarrow \\ \mathcal{V}^{\varnothing} \circ \overline{\mathcal{H}}^{\mathbf{1},\mathbf{1}}_{M,M'} & \\ \mathrm{InsR}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\leftrightarrow}\right) \circ \overline{\mathcal{H}}^{\mathbf{1},+}_{M,M'} & \\ \mathrm{DelR}\left(\mathcal{H}_{\mathsf{D},\leftrightarrow,\varnothing}\right) \circ \overline{\mathcal{H}}^{+,\mathbf{1}}_{M,M'} & \end{cases} \tag{7}$$

For every $M, M' \in \{\varnothing, \leftrightarrow, \rightarrow\}$ and $i, j \in \{\mathbf{1}, +\}$:

$$\overline{\mathcal{H}}^{i,j}_{M,M'} = \mathcal{H}_{\mathsf{I|D},\alpha(M),\alpha(M')} \oplus \begin{cases} \mathcal{F}_I & \text{if } M = \varnothing \text{ and } M' = \rightarrow \\ \mathcal{F}_I & \text{if } M = \varnothing, M' = \leftrightarrow \text{ and } j = + \\ \mathcal{F}_D & \text{if } M = \rightarrow \text{ and } M' = \varnothing \\ \mathcal{F}_D & \text{if } M = \leftrightarrow,\ M' = \varnothing \text{ and } i = + \\ \varnothing & \text{otherwise} \end{cases} \tag{8}$$

where $\alpha(\varnothing) = \varnothing$ and $\alpha(\leftrightarrow) = \alpha(\rightarrow) = \rightarrow$.

Figure 3. A context-free grammar for $\mathcal{A}$, a representative set of all tree alignments.

Theorem 2. *The set of supertrees $\mathcal{A}$ generated by the grammar* (1)-(8) *is representative of the set of all tree alignments; i.e. $\mathcal{A}$ contains exactly one supertree for each equivalence class of supertrees.*

Example 3. *Let us show how to obtain the alignment of $\mathcal{A}$ that corresponds to the supertree A_1 of Figure 1 with the grammar* (1)-(8)*. We first transform $\mathcal{A}$ into $\mathcal{V}^{\varnothing}$ via the rule* (1)*, then $\mathcal{V}^{\uparrow}$ via* (4)*, which finally becomes* $\mathrm{MatchR}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\varnothing}, A, A\right)$ *via* (5)*. Then $\mathcal{H}_{\mathsf{I|D},\varnothing,\varnothing}$ is transformed to $\mathcal{T}_I \circ \mathcal{T}_I \circ \mathcal{H}_{\mathsf{I|D},\varnothing,\varnothing}$ (transition* (7) *twice), where each $\mathcal{T}_I$ corresponds to the one-vertex insertion tree labeled by C (with the transitions $\mathcal{T}_I \rightarrow \mathrm{InsR}\left(\mathcal{F}_I, C\right) \rightarrow \mathrm{InsR}\left(\textit{empty superforest}, C\right)$). Then $\mathcal{H}_{\mathsf{I|D},\varnothing,\varnothing}$ becomes $\mathcal{V}^{\varnothing} \circ \overline{\mathcal{H}}^{\mathbf{1},\mathbf{1}}_{\varnothing,\varnothing}$ (transition* (7)*). On one hand, $\overline{\mathcal{H}}^{\mathbf{1},\mathbf{1}}_{\varnothing,\varnothing}$ becomes $\mathcal{H}_{\mathsf{I|D},\varnothing,\varnothing}$ (transition* (8)*) then the empty superforest (transition* (7)*). On the other hand, $\mathcal{V}^{\varnothing}$ is transformed*

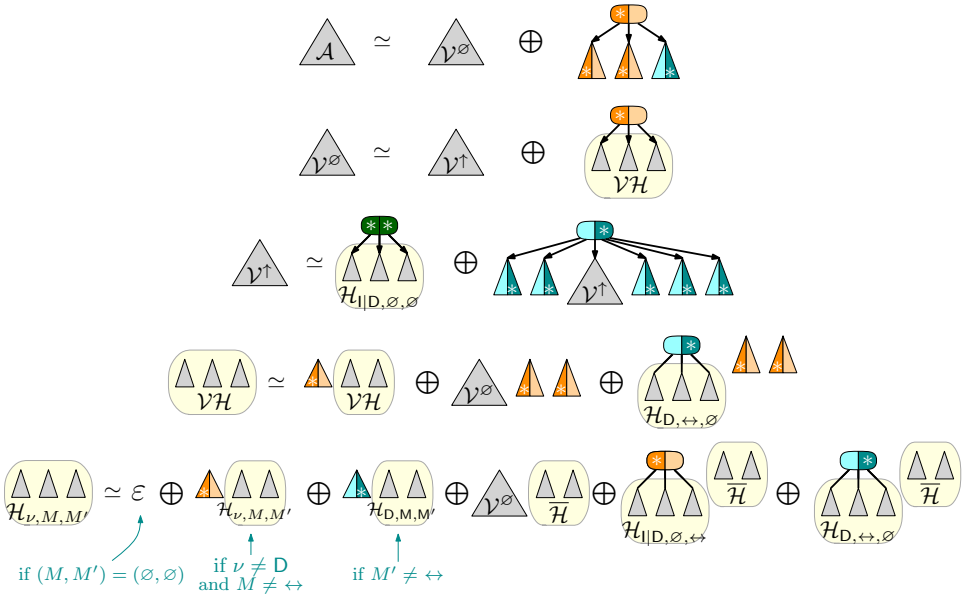

Figure 4. A schematic illustration of the grammar for tree alignments.

into $\mathcal{V}^{\uparrow}$ *then* $\mathrm{DelR}\left(\mathcal{F}_D \circ \mathcal{V}^{\uparrow} \circ \mathcal{F}_D, G\right)$. *The first* $\mathcal{F}_D$ *is the one-vertex deletion tree labeled by* U, *the second one is the empty forest. Finally, we have the transitions* $\mathcal{V}^{\uparrow} \to \mathrm{MatchR}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\varnothing}, U, A\right) \to \mathrm{MatchR}\left(\textit{empty superforest}, U, A\right)$.

The key ingredient to prove Theorem 2 stems from the following (semantic) properties for the classes of supertrees and forests that appear in the grammar:

(1) Supertrees in $\mathcal{T}_I$ (resp. $\mathcal{T}_D$) contain only insertion (resp. deletion) vertices.
(2) $\mathcal{F}_I$ (resp. $\mathcal{F}_D$) is the set of superforests formed by supertrees of $\mathcal{T}_I$ (resp. $\mathcal{T}_D$).
(3) For $\mu \in \{\varnothing, \uparrow\}$, $\mathcal{V}^{\mu}$ is representative of the set of alignments A with at least one match, such that, if $\mu = \uparrow$, then the root of $\pi_1(A)$ is matched.
(4) $\mathcal{VH}$ is representative of the set of forest alignments A with at least one match, such that $\pi_2(A)$ is a tree.
(5) For $\nu \in \{\mathsf{I|D}, \mathsf{D}\}$ and $(M, M') \in \{\varnothing, \leftrightarrow, \rightarrow\}^2$, $\mathcal{H}_{\nu,M,M'}$ is representative of the set of superforests A such that
   - if $\pi_1(A) \neq \varnothing$ and $\nu = \mathsf{D}$, then the first tree of $\pi_1(A)$ is matched in A;
   - if $M = \rightarrow$, then the last tree of $\pi_1(A)$ is matched in A (so $\pi_1(A) \neq \varnothing$);
   - if $M' = \rightarrow$, then the last tree of $\pi_2(A)$ is matched in A (so $\pi_2(A) \neq \varnothing$);
   - if $M = \leftrightarrow$, then the first and last trees in $\pi_1(A)$ are matched in A (so $\pi_1(A)$ has at least two trees);
   - if $M' = \leftrightarrow$, then the first and last trees in $\pi_2(A)$ are matched in A (so $\pi_2(A)$ has at least two trees).

(6) For $i,j \in \{\mathbf{1},+\}^2$, $\overline{\mathcal{H}}^{i,j}_{M,M'}$ is representative of superforests A' such that
- there exists a superforest A such that $A \circ A' \in \mathcal{H}_{\mathrm{D},M,M'}$;
- if $i = \mathbf{1}$ (resp. $+$), $\pi_1(A)$ is a tree (resp. a forest with at least two trees);
- if $j = \mathbf{1}$ (resp. $+$), $\pi_2(A)$ is a tree (resp. a forest with at least two trees).

These properties can be verified recursively through a tedious analysis of the grammar, which will be formally proved in Section 5. The important point for the unambiguity is the fact that the unions are indeed disjoint, which is generally straightforward.

Remark 1. *For sequences alignments, a grammar generating a representative set of sequence alignments can be easily adapted from the grammar generating all sequences over Σ_a, e.g. by preventing any occurrence to immediately precede an insertion. In the case of trees, the two-dimensional nature of the objects seems to forbid such a simple characterization, and seem to intrinsically mandate intricate combinatorial constructs/grammars. Note however, that our grammar, while complex, remains amenable to efficient computations, as shown in the upcoming Section 4.*

4. Applications

4.1. *Enumeration and uniform random generation of tree alignments*

From an enumerative point of view, numerous exact and asymptotic results can be automatically derived from classes of objects which are generated by context-free grammars. This section presents some of these results for tree alignments, given the grammar we described in the previous subsection.

For a family $\mathcal{F}$ of superforests, we define a *bivariate ordinary generating function*

$$F(t,z) = \sum_{n\geq 0,\, k\geq 0} f_{n,k}\, t^n\, z^k$$

where $f_{n,k}$ is the number of superforests in $\mathcal{F}$ of size n with k matches. For example, the first terms of the generating function of the tree alignments are given by

$$T(t,z) = m^2t^2 + m^2t^2z + 2\,m^3t^3 + 4\,m^3t^3z + 5\,m^4t^4 + 16\,m^4t^4z + m^4t^4z^2 + \dots$$

where m is the size of the alphabet. The $4m^3t^3z$ term means there exist $4m^3$ tree alignments between two trees of cumulative size 3 (so one must have 1 node, the other one 2 nodes) with exactly one match.

Theorem 4. *The generating function $T(t,z)$ of tree alignments on an alphabet of size m, whose size and number of matches are marked by t and z respectively, is given by:*

$$T(t,z) = \left(\boldsymbol{t}^2 + \boldsymbol{t}^2 z + 2\boldsymbol{t}^2 z\left(\frac{1}{2\sqrt{1-4\,\boldsymbol{t}}} - \frac{1}{2}\right)\right) F(t,z), \tag{9}$$

where $\boldsymbol{t} = mt$ *and* $F(t, z)$*, the generating function of forests, is the positive solution of*

$$(\boldsymbol{t}zC(\boldsymbol{t})^2 - \boldsymbol{t}^2C(\boldsymbol{t})^2 + 2\boldsymbol{t})F(t,z)^2 + (\boldsymbol{t}^2C(\boldsymbol{t})^4 - 2\boldsymbol{t}C(\boldsymbol{t})^2 - 1)F(t,z) + C(\boldsymbol{t})^2 = 0 \quad (10)$$

with $C(t) = 1 - \sqrt{1-4t}/2t$ *being the generating function of Catalan numbers.*

Proof. First, remark that for the families of alignments by our grammars, we can assume without loss of generality that $m = 1$. Indeed, none of these classes enforce any restriction on the labeling of the supertrees, which means that we can always replace a label with another one. This implies that $T(t/m, z)$ and $F(t/m, z)$ do not depend on m.

We prove this theorem using *symbolic method* [11]. This theory classically translates the specification described by Eqs. (1)-(8) into a system of functional equations relating the generating functions of various subclasses of objects, *e.g.* subsets of supertrees and forests. To that purpose, classes of objects are replaced by their generating function, disjoint unions (resp. concatenations) of two sets of supertrees are replaced by additions (resp. multiplications) of their generating functions, the addition of a root translates into a multiplication by a monomial t^2z (resp. t) if the root represents a match (resp. insertion/deletion), and the empty superforest and set translate into $z^0 = 1$ and 0 respectively.

The grammar is context-free, so the resulting system is algebraic. In theory, the generating function of tree alignments, which counts the alignments generated from $\mathcal{A}$ in the grammar of Fig. 3, can be straightforwardly extracted from this system (with a formal solver software, like *maple*). However in practice, because the functional system is quite involved, we did not manage to obtain an explicit expression for $T(t, z)$ with this method.

Fortunately, by using symmetries between the generating functions[b], it is possible to find a simple equation or the generating function of **forest** alignments, namely Equation (10).

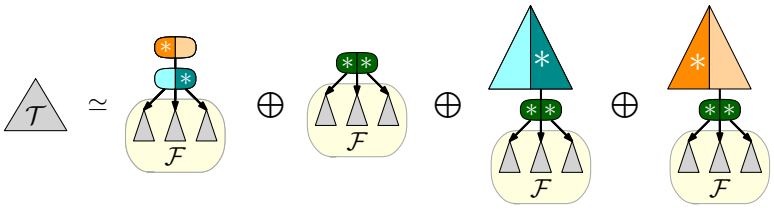

Figure 5. Expressing tree alignments in terms of forest alignments.

[b]For instance, the generating function of the superforests generated from $\mathcal{H}_{\mathsf{I|D},M,M'}$ is the same as the one generated from $\mathcal{H}_{\mathsf{I|D},M',M}$.

Size	0	1	2	3	4	5	6	7	8	9
#Tree alignments	0	0	2	6	22	88	370	1,612	7,232	33,304
#Forest alignments	1	2	6	22	90	394	1,808	8,596	42,030	210,234

Table 1. Number of alignments for sizes up to 9.

The expression for $T(t,z)$ is finally obtained by decomposing tree alignments in terms of forest alignments. This decomposition, which provides Equation (9), is illustrated by Figure 5. Let us explain its principle. Given a tree alignment A between two trees S and T, there are four possibilities. (a) The root node of S comes from an insertion of A, and the root node of T comes from a deletion. In that case, A is equivalent to some forest alignment to which we have added at the root a deletion vertex then an insertion vertex with the same labels as the roots of T and S. (b) The root nodes of S and T come from a match of A. Then the alignment A begins by a match vertex, and its children form an unconstrained forest alignment. (c) The root node of S comes from a match vertex mv of A, but the root node of T does not. Then, in A, every vertex which is not a descendant of mv must be a deletion vertex. Indeed, every insertion or match vertex must be under mv since mv induces the root of S. Moreover, the children of mv in A form an unrestrained forest alignment. (d) The root node of T comes from a match vertex mv of A, but the root node of S does not. It is the symmetric case of (c).

Let us prove Equation (9) by enumerating the tree alignments from the contributions (a), (b), (c), (d). The cases (a) and (b) differ from a forest alignment by an addition of root(s), so are respectively counted by $t^2F(t,z)$ and $zt^2F(t,z)$. The cases (c) and (d) are symmetric so are counted by the same numbers. The alignments in question are trees to which we have inserted a tree formed by a match vertex and a forest alignment. Given a tree T of size k, there are $(2k-1)$ ways to insert an other tree inside T. One can check that the generating function of plane trees of size k times $(2k-1)$ is

$$\frac{1}{2\sqrt{1-4\,t}} - \frac{1}{2}.$$

At the end, the tree alignments coming from (3) or (4) are counted by

$$\left(\frac{1}{2\sqrt{1-4\,t}} - \frac{1}{2}\right) t^2\, z\, F(z,t).$$

Summing all the contributions give Equation (9). □

Solving the quadratic equation (10) leads to an explicit formula for F (and hence T). The generating function of tree alignments is therefore

$$T(t,z) = \frac{\left(z+\sqrt{1-4\,\boldsymbol{t}}\right)\left(2+8\,\boldsymbol{t}^2-(2-8\,\boldsymbol{t})\sqrt{1-4\,\boldsymbol{t}}-12\,\boldsymbol{t}+2\sqrt{2}R\right)}{4\sqrt{1-4\,\boldsymbol{t}}\left(1-z-(1-z)\sqrt{1-4\,\boldsymbol{t}}-6\,\boldsymbol{t}+2\,z\boldsymbol{t}\right)} \tag{11}$$

where

$$R = \sqrt{\left(1 - 8\,\boldsymbol{t} + 16\,\boldsymbol{t}^2 - 4\,z\boldsymbol{t}^2\right)\left(2\,\boldsymbol{t}^2 + (2\,\boldsymbol{t} - 1)\sqrt{1 - 4\,\boldsymbol{t}} + 1 - 4\,\boldsymbol{t}\right)}.$$

This explicit expression allows to compute the first coefficients as shown in Table 1. More relevantly, it can be used to derive asymptotic estimates using *transfer theorems* [11] .

Theorem 5. *The number of tree alignments of size n is asymptotically equivalent to $\kappa \times n^{-3/2} \times 6^n$, where $\kappa = \sqrt{2}(3 - \sqrt{3})/(24\sqrt{\pi})$.*

Remind that there are asymptotically

$$\frac{4^n}{4\sqrt{\pi} \cdot n\sqrt{n}}$$

ordered pairs of labeled rooted trees of cumulated size n, and that each tree alignment A of $\mathcal{A}$ corresponds to exactly an alignment for one pair of such trees $(\pi_1(A), \pi_2(A))$. We can thus deduce from the previous proposition that, on average, a pair of trees of cumulated size n admits $\kappa' \times 1.5^n$ tree alignments, where $\kappa' = \sqrt{2}(3 - \sqrt{3})/6 \simeq 0.299$.

Corollary 1. *The average number of tree alignments for a random pair of trees of cumulated size n is $\kappa' \times 1.5^n$, where $\kappa' = \sqrt{2}(3 - \sqrt{3})/6$.*

Other techniques of analytic combinatorics can be used to characterize the distribution of the number of matches in a random tree alignment. A direct application of Theorem IX.12 from [11] indeed gives the following.

Proposition 2. *Let m_n be the random variable that counts the number of matches in a uniformly-drawn random tree alignment. The variable m_n follows a Normal law of mean $\mathbb{E}(m_n) \sim n/6$ and variance $\mathbb{V}(m_n) \sim n/6$.*

Finally, the grammar described in Figure 3 immediately yields polynomial-time random generation algorithms based on the recursive method [24, 9] or Boltzmann sampling [8].

Proposition 3. *The uniform random generation of $k > 0$ tree alignments of length n can be performed using:*

(1) $\mathcal{O}(k \cdot n \log n)$ arithmetic operations after a precomputation involving $\Theta(n)$ operations;
(2) $\mathcal{O}(k \cdot n^2)$ expected time after a precomputation in $\Theta(1)$ time.

Proof. Result 1 follows from a direct application of the symbolic method [9]. For each non-terminal $\mathcal{M}$, the number m_n of trees/forests of size n is precomputed.

Instead of using explicit convolution products, which would require $\Theta(n^2)$ operations, this precomputation can be performed in $\Theta(n)$ arithmetic operations. To that purpose, one uses the fact that the coefficients of holonomic generating functions, a superset of algebraic generating functions, admit linear recurrences with polynomial coefficients. The generation uses precomputed numbers to ensure uniformity, using a Boustrophedon order to efficiently investigate suitable factorizations in the case of products, leading to a worst-case complexity of generation in $\mathcal{O}(n \log n)$ failed comparisons.

Result 2 is a generic application of Boltzmann sampling in the context of unlabeled combinatorial structures [10]. Indeed, it can be verified that the grammar is strongly connected, with the exception of the $\mathcal{T}_I$ and $\mathcal{T}_D$ non-terminals. However, the generating functions associated with these non-terminals admit a dominant singularity at $\rho_{I|D} = 1/4m$, having modulus strictly larger than that of the main connected component, and therefore constitute a case of subcritical composition [11]. The singularity of tree alignments therefore remains of the ubiquitous square-root type, and the proof of Flajolet, Fusy and Pivoteau [10] can therefore be easily adapted to yield the claimed average-case complexity for an exact-size Boltzmann sampler. □

4.2. *Sampling alignments between two given trees*

We now consider two *fixed* trees S and T, and address the task of sampling a tree alignment A such that $\pi_1(A) = S$ and $\pi_2(A) = T$, with respect to the Gibbs-Boltzmann probability distribution. This can be used, for instance, to assess the stability of a predicted alignment. We refer the interested reader to Section 1 for further motivations and examples of applications in applied contexts.

Previously published dynamic programming algorithms for computing tree alignments were not suited to this approach due to the fact that their underlying (implicit) supertrees grammars were provably ambiguous, and thus not amenable to our algebraic approach.

Definitions and problem statement. Let $\mathcal{T}_{S,T}$ be the set of all supertrees A such that $\pi_1(A) = S$ and $\pi_2(A) = T$, and $\mathcal{A}_{S,T}$ be a representative set of $\mathcal{T}_{S,T}$. In other words, $\mathcal{A}_{S,T}$ can be interpreted as the set of all alignments between S and T. For any supertree $A \in \mathcal{T}_{S,T}$, we define its *edit score* $s(A)$ as the sum of the number of insertions, deletions and matches (x, y) such that $x \neq y$. Note that this definition can be trivially extended to any edit scoring system that is a positive linear combination of the numbers of insertions, deletions and matches.

For a given positive constant $k\theta$, the *partition function* $Z_{S,T}$ of $\mathcal{A}_{S,T}$ and the

Gibbs-Boltzmann probability $\Pr(A)$ of an alignment $A \in \mathcal{A}_{S,T}$ are defined as

$$Z_{S,T} = \sum_{A \in \mathcal{A}_{S,T}} e^{-s(A)/k\theta} \quad \text{and} \quad \Pr(A) = \frac{e^{-s(A)/k\theta}}{Z_{S,T}}.$$

When $k\theta$ tends to 0, this distribution tends to the uniform distribution over supertrees of minimum edit score, while, when $k\theta$ tends to $+\infty$, it tends toward the uniform distribution over $\mathcal{A}_{S,T}$.

We consider the following problem: given two trees S and T, and a positive constant $k\theta$, sample an alignment between S and T under the Gibbs-Boltzmann probability distribution. This problem generalizes the classic combinatorial optimization problem of computing a tree alignment between S and T having minimum edit score.

Our solution builds on a deeply-rooted connection between combinatorial specifications and dynamic programming, using the approach described, among others, in [18] for RNA folding. We begin by adapting the grammar of Section 3 into a specialized grammar for the alignment space $\mathcal{A}_{S,T}$, by adding constraints arising from the two given trees S and T. Next, we transform the unambiguous grammar generating the solution space into dynamic programming algorithms for computing the partition function and sampling representative supertrees.

A grammar for $\mathcal{A}_{S,T}$. In order to guarantee that each supertree A indeed aligns two input trees S and T (namely $\pi_1(A) = S$ and $\pi_2(A) = T$), we need to restrict which rules in the grammar can be used, conditionally to which trees and forests are currently being generated. To that purpose, we introduce, for each set $\mathcal{S}$ in the previous grammar, an indexed version $\mathcal{S}_{[u,v]}$ which denotes the restriction of $\mathcal{S}$ to alignments between u and v two forests in S and T.

Slightly abusing previous notations, we denote by $a(u)$ the tree whose root is a vertex a and whose (forest of) children is u. Finally, for every tree/forest X, $\mathrm{InsT}(X)$ (resp. $\mathrm{DelT}(X)$) represents the supertree/superforest obtained from X by inserting (resp. deleting) each of its elements. If X is empty, $\mathrm{InsT}(X)$ and $\mathrm{DelT}(X)$ denote the empty superforest. The grammar for $\mathcal{A}_{S,T}$ is described in Fig. 6.

Theorem 6. *Let S and T be non-empty trees. The set of supertrees $\mathcal{A}_{S,T}$, unambiguously generated by grammar* (12)–(19)*, is representative of $\mathcal{T}_{S,T}$ the tree alignments of S and T.*

A formal proof of this result, omitted for the sake of simplicity, could be obtained by carefully checking, for each of the production rules in the grammar of Figure 3, that only the set of alignment consistent with the given (sub)-tree/forest can be generated, and that the recursive calls pertain to the suitable subtrees and forests.

Computing the generating function. The grammar defined by Equations (12)–(19) is a decomposition scheme for the alignments between S and T. It can easily

$$\mathcal{A}_{\substack{S,T \\ S\equiv r_S(X_S)}} = \mathcal{V}^{\varnothing}[S,T] \oplus \mathrm{InsR}\left(\mathrm{InsT}(X_S)\circ\mathrm{DelT}(T), r_S\right) \tag{12}$$

$$\mathcal{V}^{\varnothing}[a(u),b(v)] = \mathcal{V}^{\uparrow}[a(u),b(v)] \oplus \mathrm{InsR}\left(\mathcal{VH}[u,b(v)],a\right) \tag{13}$$

$$\mathcal{V}^{\uparrow}[a(u),b(v)] = \bigoplus \begin{cases} \mathrm{MatchR}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\varnothing}[u,v],a,b\right) \\ \bigoplus\limits_{Y\circ c(w)\circ Y'=v} \mathrm{DelR}\left(\mathrm{DelT}(Y)\circ\mathcal{V}^{\uparrow}[a(u),c(w)]\circ\mathrm{DelT}(Y'),b\right) \end{cases} \tag{14}$$

$$\mathcal{VH}[\varnothing,b(v)] = \varnothing \tag{15}$$

$$\mathcal{VH}[a(u)\circ X,b(v)] = \bigoplus \begin{cases} \mathrm{InsT}(a(u))\circ\mathcal{VH}[X,b(v)] \\ \bigoplus\limits_{\substack{X'\circ X''=a(u)\circ X \\ |X'|\geq 2}} \mathrm{DelR}\left(\mathcal{H}_{\mathsf{I|D},\leftrightarrow,\varnothing}[X',v],b\right)\circ\mathrm{InsT}(X'') \\ \mathcal{V}^{\varnothing}[a(u),b(v)]\circ\mathrm{InsT}(X) \end{cases} \tag{16}$$

For every ν, M, M' with $\nu \in \{\mathsf{I|D},\mathsf{D}\}$ and $M, M' \in \{\varnothing,\leftrightarrow,\rightarrow\}$:

$$\mathcal{H}_{\nu,M,M'}[X,\varnothing] = \begin{cases} \mathrm{InsT}(X) & \text{if } (M,M') = (\varnothing,\varnothing), \\ \varnothing & \text{otherwise,} \end{cases} \tag{17}$$

$$\mathcal{H}_{\nu,M,M'}[\varnothing,Y] = \begin{cases} \mathrm{DelT}(Y) & \text{if } (M,M') = (\varnothing,\varnothing), \\ \varnothing & \text{otherwise,} \end{cases} \tag{18}$$

$\mathcal{H}_{\nu,M,M'}[a(u)\circ X, b(v)\circ Y] =$

$$\bigoplus \begin{cases} \mathrm{InsT}(a(u))\circ\mathcal{H}_{\nu,M,M'}[X,b(v)\circ Y] & \text{if } \nu \neq \mathsf{D} \text{ and if } M \neq \leftrightarrow, \\ \mathrm{DelT}(b(v))\circ\mathcal{H}_{\mathsf{D},M,M'}[a(u)\circ X,Y] & \text{if } M' \neq \leftrightarrow, \\ \mathcal{V}^{\varnothing}[a(u),b(v)]\circ\mathcal{H}_{\mathsf{I|D},\alpha(M,X),\alpha(M',Y)}[X,Y] & \\ \bigoplus\limits_{\substack{Y'\circ Y''=b(v)\circ Y \\ |Y'|\geq 2}} \mathrm{InsR}\left(\mathcal{H}_{\mathsf{I|D},\varnothing,\leftrightarrow}[u,Y'],a\right)\circ\mathcal{H}_{\mathsf{I|D},\alpha(M,X),\alpha(M',Y'')}[X,Y''] & \\ \bigoplus\limits_{\substack{X'\circ X''=a(u)\circ X \\ |X'|\geq 2}} \mathrm{DelR}\left(\mathcal{H}_{\mathsf{D},\leftrightarrow,\varnothing}[X',v],b\right)\circ\mathcal{H}_{\mathsf{I|D},\alpha(M,X''),\alpha(M',Y)}[X'',Y] & \end{cases} \tag{19}$$

where $\alpha(\varnothing,X) = \varnothing$ and $\alpha(\leftrightarrow,X) = \alpha(\rightarrow,X) = \begin{cases} \varnothing & \text{if } X = \varnothing, \\ \rightarrow & \text{otherwise.} \end{cases}$

Figure 6. A grammar for $\mathcal{A}_{S,T}$, a representative set of all tree alignments between two input trees S and T.

be transformed into an algorithm for computing the partition function $Z_{S,T}$. Indeed, $Z_{S,T}$ is simply a weighted sum over all possible supertrees of $\mathcal{A}_{S,T}$ the set generated by the grammar. Indeed, consider the set of numerical equations obtained by applying classic syntactical transforms [4]. The conversion is clarified in the context of our grammar for tree alignments in Table 2.

Type	Rule	Supertree Set	Partition fun.
Empty set	$\mathcal{P} = \varnothing$	$\mathcal{A}_\mathcal{P} = \varnothing$	$Z_\mathcal{P} = 0$
Tree S ins.	$\mathcal{P} = \mathrm{InsT}(S)$	$\mathcal{A}_\mathcal{P} = \{\mathrm{InsT}(S)\}$	$Z_\mathcal{P} = e^{-\frac{\lvert S\rvert}{k\theta}}$
Tree T del.	$\mathcal{P} = \mathrm{DelT}(T)$	$\mathcal{A}_\mathcal{P} = \{\mathrm{DelT}(S)\}$	$Z_\mathcal{P} = e^{-\frac{\lvert T\rvert}{k\theta}}$
Root ins.	$\mathcal{P} = \mathrm{InsR}(\mathcal{Q}, a)$	$\mathcal{A}_\mathcal{P} = \{\mathrm{InsR}(t, a)\}_{t\in\mathcal{A}_\mathcal{Q}}$	$Z_\mathcal{P} = e^{-\frac{1}{k\theta}} \cdot Z_\mathcal{Q}$
Root del.	$\mathcal{P} = \mathrm{DelR}(\mathcal{Q}, b)$	$\mathcal{A}_\mathcal{P} = \{\mathrm{DelR}(t, b)\}_{t\in\mathcal{A}_\mathcal{Q}}$	$Z_\mathcal{P} = e^{-\frac{1}{k\theta}} \cdot Z_\mathcal{Q}$
Roots match	$\mathcal{P} = \mathrm{MatchR}(\mathcal{Q}, a, b)$	$\mathcal{A}_\mathcal{P} = \{\mathrm{MatchR}(t, a, b)\}_{t\in\mathcal{A}_\mathcal{Q}}$	$Z_\mathcal{P} = e^{-\frac{1_{a\neq b}}{k\theta}} \cdot Z_\mathcal{Q}$
Union	$\mathcal{P} = \mathcal{Q} \oplus \mathcal{R}$	$\mathcal{A}_\mathcal{P} = \mathcal{A}_\mathcal{P} \cup \mathcal{A}_\mathcal{R}$	$Z_\mathcal{P} = Z_\mathcal{Q} + Z_\mathcal{R}$
Product	$\mathcal{P} = \mathcal{Q} \circ \mathcal{R}$	$\mathcal{A}_\mathcal{P} = \mathcal{A}_\mathcal{P} \times \mathcal{A}_\mathcal{R}$	$Z_\mathcal{P} = Z_\mathcal{Q} \times Z_\mathcal{R}$

Table 2. Conversion scheme from supertree constructs and grammar rules to partition function recurrences in the grammar of Figure 6. Binary operators and isolated terminal rules are assumed without loss of generality, as any grammar can be transformed into a binary one through the introduction of dedicated intermediate non-terminals.

Proposition 4. *The equations obtained by applying the constructs of Table 2 to the grammar of Figure 6 computes the partition functions associated with each non-terminal.*

Proof. We detail and justify the several cases summarized in Table 2 for each type of rule $\mathcal{P}$. For the empty set, the partition function is trivially $\sum_{A\in\varnothing} e^{-s(A)/k\theta} = 0$. The insertion (resp. deletion) of a tree T produces a single supertree, whose topology mimics that of T, but whose nodes are replaced by insertion (resp. deletion) nodes. This supertree has an edit score of $|T|$ and is the sole contributor to the partition function, thus we get $\sum_{A\in\mathcal{A}_\mathcal{P}} e^{-s(A)/k\theta} = e^{-|T|/k\theta}$ as stated.

Appending an insertion (resp. deletion) node at the top of a supertree generated from $\mathcal{Q}$ increases its edit score by 1. Therefore one has

$$Z_\mathcal{P} = \sum_{A\in\mathcal{A}_\mathcal{P}} e^{-\frac{s(A)}{k\theta}} = \sum_{A\in\mathcal{A'}_\mathcal{Q}} e^{-\frac{s(A')+1}{k\theta}} = e^{-\frac{1}{k\theta}} \sum_{A\in\mathcal{A'}_\mathcal{P}} e^{-\frac{s(A')}{k\theta}} = e^{-1/k\theta} \cdot Z_\mathcal{Q}$$

in the root insertion/deletion cases. The case of mismatches (MatchR nodes with $a \neq b$) is identical, leading to the same recurrence. Finally, appending a match (MatchR with $a = b$) node to a tree in $\mathcal{A}_\mathcal{Q}$ does not change its edit score, thus $Z_\mathcal{P} = Z_\mathcal{Q}$.

In the case of (disjoint) union rules, one has

$$Z_\mathcal{P} = \sum_{A\in\mathcal{A}_\mathcal{P}} e^{-\frac{s(A)}{k\theta}} = \sum_{A\in\mathcal{A}_\mathcal{Q}\cup\mathcal{A}_\mathcal{R}} e^{-\frac{s(A)}{k\theta}} = \sum_{A\in\mathcal{A}_\mathcal{Q}} e^{-\frac{s(A)}{k\theta}} + \sum_{A\in\mathcal{A}_\mathcal{R}} e^{-\frac{s(A)}{k\theta}} = Z_\mathcal{Q} + Z_\mathcal{R}.$$

Finally, the product case is provably correct since

$$Z_\mathcal{P} = \sum_{A',A''\in\mathcal{A}_\mathcal{Q}\times\mathcal{A}_\mathcal{R}} e^{-\frac{s(A')+s(A'')}{k\theta}} = \sum_{A'\in\mathcal{A}_\mathcal{Q}} e^{-\frac{s(A')}{k\theta}} \sum_{A''\in\mathcal{A}_\mathcal{R}} e^{-\frac{s(A'')}{k\theta}} = Z_\mathcal{Q} \times Z_\mathcal{R}.$$

Type	Rule	Generator
Empty set	$\mathcal{P} = \varnothing$	$\Gamma_{\mathcal{P}} :=$ **return** `Error`
Tree S ins.	$\mathcal{P} = \mathrm{InsT}(S)$	$\Gamma_{\mathcal{P}} :=$ **return** $\mathrm{InsT}(S)$
Tree T del.	$\mathcal{P} = \mathrm{DelT}(T)$	$\Gamma_{\mathcal{P}} :=$ **return** $\mathrm{DelT}(T)$
Root ins.	$\mathcal{P} = \mathrm{InsR}(\mathcal{Q}, a)$	$\Gamma_{\mathcal{P}} :=$ **return** $\mathrm{InsR}(\Gamma_{\mathcal{Q}}, a)$
Root del.	$\mathcal{P} = \mathrm{DelR}(\mathcal{Q}, b)$	$\Gamma_{\mathcal{P}} :=$ **return** $\mathrm{DelR}(\Gamma_{\mathcal{Q}}, b)$
Roots match	$\mathcal{P} = \mathrm{MatchR}(\mathcal{Q}, a, b)$	$\Gamma_{\mathcal{P}} :=$ **return** $\mathrm{MatchR}(\Gamma_{\mathcal{Q}}, a, b)$
Union	$\mathcal{P} = \mathcal{Q} \oplus \mathcal{R}$	$\Gamma_{\mathcal{P}} :=$ **return** $\begin{cases} \Gamma_{\mathcal{Q}} & \text{with probability } Z_{\mathcal{Q}}/Z_{\mathcal{P}} \\ \Gamma_{\mathcal{R}} & \text{otherwise} \end{cases}$
Product	$\mathcal{P} = \mathcal{Q} \circ \mathcal{R}$	$\Gamma_{\mathcal{P}} :=$ **return** $\Gamma_{\mathcal{Q}} \circ \Gamma_{\mathcal{R}}$

Table 3. Gibbs-Boltzmann generators associated with the rules of Figure 6.

The correctness of our whole transform follows from an induction on the maximum number of production rules needed to generate a given supertree. □

Gibbs-Bolzmann sampling algorithm. Now we know the value of the partition function of the set of supertrees associated with each non-terminal of the image grammar. This can be used to define an algorithm to sample supertrees from $\mathcal{A}_{S,T}$ under the Gibbs-Boltzmann distribution, following a weighted instance of the *recursive* method for random generation [24, 4], whose atomic generators are summarized in Table 3.

Proposition 5. *Let S and T be two trees. The sampling algorithm adapted from grammar* (12)–(19) *using the generators of Table 3 samples a supertree/forest from $\mathcal{A}_{S,T}$ under the Gibbs-Boltzmann distribution.*

Proof. We establish the correctness of the algorithm by induction on the (finite) number of production rules used to generate a supertree. In the case of whole tree insertion/deletion, the Boltzmann distribution is restricted to a single element, so a deterministic generation of this element is consistent with the Boltzmann distribution.

In the case of a root insertion (resp. deletion), the induction hypothesis ensures that Γ_Q generates any supertree $A \in \mathcal{A}_{\mathcal{Q}}$ with Boltzmann-Gibbs probability. Moreover, we have

$$\frac{e^{-\frac{s(A)}{k\theta}}}{Z_{\mathcal{Q}}} = \frac{e^{-\frac{1}{k\theta}} e^{-\frac{s(A)}{k\theta}}}{e^{-\frac{1}{k\theta}} Z_{\mathcal{Q}}} = \frac{e^{-\frac{s(\mathrm{InsR}(A,a))}{k\theta}}}{Z_{\mathcal{P}}},$$

so $\Gamma_{\mathcal{P}}$ indeed generates a supertree under the Bolzmann-Gibbs distribution on $\mathcal{A}_{\mathcal{P}}$. The correctness of the MatchR generator follows from a similar argument.

In the product case, the induction hypothesis implies that the trees generated by $\Gamma_{\mathcal{P}}$ and $\Gamma_{\mathcal{Q}}$ follow the Bolzmann-Gibbs distributions over $\mathcal{A}_{\mathcal{Q}}$ and $\mathcal{A}_{\mathcal{R}}$ respectively. It follows that the probability of a superforest $A' \circ A'', A', A'' \in \mathcal{A}_{\mathcal{Q}} \times \mathcal{A}_{\mathcal{R}}$, being generated by $\Gamma_{\mathcal{P}}$ is

$$\frac{e^{-\frac{s(A')}{k\theta}}}{Z_{\mathcal{Q}}} \times \frac{e^{-\frac{s(A'')}{k\theta}}}{Z_{\mathcal{R}}} = \frac{e^{-\frac{s(A')+s(A'')}{k\theta}}}{Z_{\mathcal{Q}} \times Z_{\mathcal{R}}} = \frac{e^{-\frac{s(A' \circ A'')}{k\theta}}}{Z_{\mathcal{P}}}$$

in which one recognizes the Boltzmann-Gibbs probability with respect to $\mathcal{A}_{\mathcal{P}}$.

Finally, union rules generate $A \in \mathcal{A}_{\mathcal{Q}}$ and $A' \in \mathcal{A}_{\mathcal{P}}$ with respective probabilities

$$\frac{Z_{\mathcal{Q}}}{Z_{\mathcal{P}}} \times \frac{e^{-\frac{s(A)}{k\theta}}}{Z_{\mathcal{Q}}} = \frac{e^{-\frac{s(A)}{k\theta}}}{Z_{\mathcal{P}}}$$

and

$$\left(1 - \frac{Z_{\mathcal{Q}}}{Z_{\mathcal{P}}}\right) \times \frac{e^{-\frac{s(A')}{k\theta}}}{Z_{\mathcal{R}}} = \left(1 - \frac{Z_{\mathcal{Q}}}{Z_{\mathcal{Q}} + Z_{\mathcal{R}}}\right) \times \frac{e^{-\frac{s(A')}{k\theta}}}{Z_{\mathcal{R}}} = \frac{Z_{\mathcal{R}}}{Z_{\mathcal{P}}} \times \frac{e^{-\frac{s(A')}{k\theta}}}{Z_{\mathcal{R}}} = \frac{e^{-\frac{s(A')}{k\theta}}}{Z_{\mathcal{P}}}.$$

Thus our generators for disjoint union rules indeed generate supertrees/forests under the Boltzmann distribution over $\mathcal{A}_{\mathcal{P}}$, and this establishes the correctness of our generators. □

Worst-case and average-case complexity analysis. The time and space complexities of the partition function computation correspond to that of the ambiguous Jiang *et al* algorithm [16].

Proposition 6. *The above algorithm generates a random, Boltzmann-Gibbs distributed, tree alignment for two trees of sizes n_1 and n_2 respectively, in worst-case complexity $\mathcal{O}(n_1 \cdot n_2 \cdot (n_1 + n_2)^2)$ time and $\Theta(n_1 \cdot n_2 \cdot (n_1 + n_2))$ space.*

Moreover, its expected complexities are in $\Theta(n_1 \cdot n_2)$ time and space for two random, uniformly distributed, input trees of size n_1 and n_2.

Sketch of proof. The worst-case analysis is highly similar to the one performed in the seminal Jiang *et al* paper [16], and we briefly remind its key argument. We focus on the computation of the partition function rather than the generation step, whose complexity is provably bounded by $\mathcal{O}((n_1 + n_2)\log(n_1 + n_2))$ using a Boustrophedon order for investigating the ways to split the subforests. Then, remark that $\mathcal{H}_{,,}$ in Eq. (19) is the only non-terminal indexed by two subforests, *i.e.* the only possible one to induce computations that would exceed the announced complexities. However, it is easy to establish that, starting from $\mathcal{A}_{S,T}$, $\mathcal{VH}$ can only be reached with indexing subforests F and F', one of which is either a prefix or a suffix of the children in S or T. It follows that there exists at most

$$2 \cdot n_1 \cdot \frac{n_2(n_2 - 1)}{2} + 2 \cdot n_2 \cdot \frac{n_1(n_1 - 1)}{2} \in \Theta\left(n_1 \cdot n_2 \cdot (n_1 + n_2)\right)$$

reachable indexing pairs for $\mathcal{VH}$, from which we get the dominant term of the space complexity. Turning to the time complexity, note that it is dominated by the computation of the convolutions in the right-hand side of Eq. (19), investigating all the possible ways to split F (resp. F') into two parts $X \circ X'$ (resp. $Y \circ Y'$). Since there exists at most $\Theta(n_1 + n_2)$ ways of performing such a split, we obtain the claimed worst-case time complexity.

The proof of the, surprisingly reasonable, average-case complexity follows from the average-case analysis, by Herrbach *et al* [13], of the Jiang *et al* algorithm [16]. Our claim results from the observation that the structure of the DP scheme underlying their algorithm is essentially the same as the ours, and only differs on the number of DP tables (by a constant factor). Therefore the arguments of Herrbach *et al* [13] for bounding the expected growth of indexing triplets can be used directly for the analysis of our alternative algorithm. □

This grammar can also be transformed into an optimization algorithm that computes the minimum edit score among all supertrees from $\mathcal{A}_{S,T}$. To do so, it suffices to replace the set-theoretic operator $\oplus$ by the minimum operator $\min$, the superforest construction operators $\circ$ by $+$, the operators InsR, DelR, MatchR respectively by the edit score associated to an insertion (1), deletion (1) and match (0 for matches, 1 for mismatches), and the empty set $\varnothing$ by $+\infty$. The asymptotic complexities given in Proposition 6 are preserved by the algebraic substitution, and the new algorithm has complexity asymptotically equal to that of the current tree alignments algorithms [16, 3].

5. Proof of Theorem 2

The purpose of this section is to prove Theorem 2, or in other words, to show the correctness of the grammar described by Figure 3.

We begin by two lemmas, whose proofs are straightforward. The first lemma implies that, given a supertree A which aligns two trees S and T, one can find another supertree, equivalent to A, whose root induces the root of S (claim which is not obvious if the root of A belongs to T). This lemma is necessary to do inductions on alignments. Remember that $A - v$ is the supertree obtained by deleting v from a superforest A.

Lemma 7. *Let A be a supertree such that $S = \pi_1(A)$ and $T = \pi_2(A)$ are trees.*

- *Assume that the root s of S is inserted in A; let $v = \pi_1^{-1}(s)$ and $a \in \Sigma$ be the label of s. Then A is equivalent to $\mathrm{InsR}\,(A - v, a)$.*
- *Assume that the root t of T is deleted in A; let $v = \pi_2^{-1}(t)$ and $b \in \Sigma$ be the label of t. Then A is equivalent to $\mathrm{DelR}\,(A - v, b)$.*

- *Assume that the roots s of of S and t of T are matched in A; let v be the root of A, $a \in \Sigma$ be the label of s and $b \in \Sigma$ be the label of t. Then A is equivalent to* $\mathrm{MatchR}\,(A - v, a, b)$.

These statements remain true if we replace $A - v$ by any supertree equivalent to $A - v$.

The next lemma allows us to split the forest alignments in pairs of independent alignments when it is possible.

Lemma 8. *Let A be a superforest which aligns two forests F and G. Assume that there exists a decomposition $F = F_1 \circ F_2$ and $G = G_1 \circ G_2$ such that there exists no match between F_1 and G_2, and no match between F_2 and G_1. Set V as the set of vertices x of A such that $\pi_1(x) \in F_1$ or $\pi_2(x) \in G_1$. We denote by A_1 the superforest A from which we have removed every vertex that does not belong in V (in any order), and by A_2 the superforest A from which we have removed every vertex of V. Then A is equivalent to $A_1 \circ A_2$.*

The superforest A_1 from the previous lemma is called the *restriction* of A to $F_1 \cup G_1$, and A_2 is called the *complement* of $F_1 \cup G_1$ in A. Note that A_1 aligns F_1 and G_1, while A_2 aligns F_2 and G_2.

We can now address the proof of the correctness of the grammar. To do so, we prove that the semantic properties described after Example 3 correspond to the transitions of the grammar described in Figure 3. More precisely, given an alignment of a specific class, we indicate how this alignment can be decomposed in terms of other classes of alignments.

Transition (1)**.** Consider a tree alignment A between two trees S and T. There are two possibilities : either A has one match, then by induction, it has a representative element in $\mathcal{V}^{\emptyset}$; or A is only composed by insertion and deletion vertices. In the last case, A is equivalent to the tree obtained from A by removing every deletion vertex where we have inserted, as the last child of the root, the tree obtained from A by removing every insertion vertex. One can check that this means that A must be of the form $\mathrm{InsR}\,(F_I \circ T_D, a)$ where F_I is a forest only made of insertions, T_D a tree only made of deletions, and a is the label of the root of S.

Transitions (2)**-**(3)**.** These are the classical grammars for trees and forests.

Transition (4)**.** Consider a tree alignment A between two trees S and T with at least one match. There are two exclusive possibilities: either the root of S is matched and in this case A has a representative element in $\mathcal{V}^{\uparrow}$; or the root of S is inserted, which means by Lemma 7 that A is equivalent to some $\mathrm{InsR}\,(A', a)$, where A' is an alignment between a forest and a tree (hence A' is equivalent to some element in $\mathcal{V}\mathcal{H}$) and a the label of the root of S.

Transition (5)**.** Let A be a tree alignment between two trees S and T such that the root of S is matched. If the root of T is also matched, then by Lemma 7, A can be put

under the form $\mathrm{MatchR}\,(A', a, b)$, where A' is some forest alignment (no constraint) and (a, b) the label of the root of A. Otherwise, the root of T is deleted. By the same lemma, A is equivalent to $\mathrm{DelR}\,(A', b)$, where b is the label of the root of S, and A' is a superforest which aligns a tree whose root is matched, and a forest. Among the supertrees formed by A', only one can contain an insertion vertex or a match vertex, since $\pi_1(A')$ must be a single tree. If B denotes this supertree, then $\pi_2(B)$ is a single tree, since the possibility that $\pi_2(B)$ is composed by several trees only occurs when the root of B is an insertion vertex (this is not possible since the root of S is matched). In other words, B is equivalent to some element in $\mathcal{V}^{\uparrow}$, hence A' is equivalent to some element in $\mathcal{F}_D \circ \mathcal{V}^{\uparrow} \circ \mathcal{F}_D$.

Transition (6)**.** Let A be an alignment between a forest F and a tree T (with at least one match). Let T_1 be the first tree of F and $\{v_1, v_2, \ldots, v_k\}$ its vertex set. If there is no matched vertex in T_1, then by Lemma 8, A is equivalent to the concatenation of the restriction of A to T_1, which is nothing but the tree T in which we have changed every vertex by an insertion vertex, and the complement of T_1 in A. The latter forest alignment is still an alignment between a forest and a tree (with at least one match), so has a representative element in $\mathcal{V}\mathcal{H}$.

Let us assume now that T_1 has at least one match. Set T_i as the last tree of F to have one matched vertex. The trees of F after T_i must be then deleted. Let A' be the alignment obtained from A by removing these trees. If $T_1 = T_i$, then A' aligns two trees (with a least a match) so is equivalent to an alignment of $\mathcal{V}^{\varnothing}$. In other terms, A has a representative element in $\mathcal{V}^{\varnothing} \circ \mathcal{F}_I$. If $T_1 \neq T_i$, then $\pi_1(A')$ has at least two trees, hence the root of A' has deletion type. Its children form an alignment between a forest whose first and the last trees have at least one match, and another forest. Therefore, A is equivalent to some element of $\mathrm{DelR}\,(\mathcal{H}_{\mathsf{D},\leftrightarrow,\varnothing}) \circ \mathcal{F}_I$.

Transition (7)**.** Let A be an alignment between two forests F and G satisfying the description (5) above, for some ν, M, M'. We suppose that A does not align two empty forests (which can only occur when $(M, M') = (\varnothing, \varnothing)$; the presence of the symbol $\leftrightarrow$ or $\rightarrow$ forces the existence of a tree in F or F'). Let S_1 and T_1 be the first trees of F and G.

If S_1 exists (i.e. F is not empty) and has no matched vertex (which is not possible if $\nu = \mathsf{D}$ or $M = \leftrightarrow$, by definition), then, by Lemma 8, A is equivalent to the composition of S_1 (transformed into a supertree by changing its vertices by insertion vertices) and the complement of A in S_1. This last alignment keeps the same constraints as A in full, so is equivalent to some element in $\mathcal{H}_{\nu,M,M'}$.

Let us suppose now that F is empty or S_1 has a matched vertex, but T_1 does not (which cannot happen when $M' = \leftrightarrow$). Then still by the same lemma, A is equivalent to the composition of T_1 as a supertree only formed by deletion vertices, and the complement of T_1 in A. Since F is empty or S_1 has a matched vertex, this superforest must be equivalent to some element in $\mathcal{H}_{\mathsf{D},M,M'}$.

We can otherwise assume that S_1 and T_1 have both matched vertices. Let S_i the

last tree of F to be matched with T_1, and T_j the last tree of G to be matched with S_1. The case where $S_1 \neq S_i$ *and* $T_1 \neq T_j$ cannot occur.

- Suppose that $S_1 = S_i$ and $T_1 = T_j$. By Lemma 8, A is equivalent to the composition of the restriction of A to $S_1 \cup T_1$, and the complement of $S_1 \cup T_1$ in A. The former is an alignment between two trees (with a least one match), so belongs to $\mathcal{V}^{\varnothing}$, and the rest belongs by definition to $\overline{\mathcal{H}}^{\mathbf{1},\mathbf{1}}_{M,M'}$.
- Suppose that $S_1 = S_i$ but $T_1 \neq T_j$. Let G' be the subforest of G formed by the successive trees of G between T_1 and T_j (included). As previously, A is equivalent to the composition of A_1, that is the restriction of A to $S_1 \cup G$, and A_2, that is the complement of $S_1 \cup G$ in A. If A_1 was composed by several supertrees, then at least the first or the last tree of A_1 would be only formed by deletion vertices, since $\pi_1(A_1) = S_1$ is a single tree. This is not possible, since the first and last trees of G must be matched. Hence, A' is a single supertree. The root of S_1 cannot be matched since at least two trees of G must be matched. Therefore the root of S_1 is inserted, and we can use Lemma 7 to show that A' is equivalent to some $\mathrm{InsR}\,(A', a)$ where a is the label of the root of S_1 and A' a forest alignment such that the first and last trees of the second forest are matched. Moreover, the superforest A'' is by definition in $\overline{\mathcal{H}}^{\mathbf{1},+}_{M,M'}$.
- The case where $T_1 = T_j$ but $S_1 \neq S_i$, is symmetric to the previous case.

Transition (8)**.** Forest alignments A' which are suffixes of larger forest alignments A are generally unrestrained forest alignments. However if A has some constraints (such as the presence of a match in the last tree in one of the two forests), A' will inherit of these constraints, except in the few cases (which are described in the transition (8)) where one of the two forests is reduced to the empty one.

6. Conclusion and discussion

Following a classical line of research in string algorithms, we introduced the notion of equivalence for tree alignments, and described a context-free grammar that generates a representative set of all possible alignments. We also showed how this grammar can be used to derive asymptotic properties of alignments, and design an efficient dynamic programming sampling algorithm for alignments between two given trees.

From an applied point of view, our results allow to sample optimal, as well as suboptimal, tree alignments for a pair of given trees under the Gibbs-Boltzmann distribution; following the program outlined in [18], we are currently using this algorithm to revisit the alignment of RNA structures.

Existence of a simplified grammar. Our proposed decomposition for tree alignments is admittedly intricate. In particular, it is more complex than the grammars

used to generate a representative set of sequence alignments. although dynamic programming for computing optimal sequences and trees alignments are very similar. This is due to the fact that it is particularly hard to characterize a representative set of tree alignments (see Remark 1). It thus remains an open problem to design a representative set of tree alignment that would be amenable to enumeration using a simpler grammar. However, it is important to remark that, for all its intricacies, our grammar leads to algorithms whose asymptotic complexities match that of pre-existing – ambiguous – optimization algorithms.

Automating the generation of DP schemes. From a theoretical point of view, we believe that tree alignments as defined in this work form an interesting combinatorial family whose properties deserve to be explored in depth. More generally, it would be interesting to characterize the conditions under which an instance-agnostic grammar, enumerating a search space, could be adapted into a decomposition for a specific instance. Such a theory, at the confluence of enumerative combinatorics and algorithmic design, could provide another principled ways to design dynamic-programming algorithms in a way that would complement Algebraic Dynamic Programming [19, 12].

On the average-case complexity of tree-alignment. A striking consequence of Proposition 6 is that, on average and up to constants, it is equally as difficult to align trees and sequences. This result is surprising since every existing DP equation for tree alignment is quadratic, while their counterpart for sequence alignment are linear. However, the observed discrepancy between the worst and average-case time complexities is mostly an artifact of the uniform random model, and probably poorly representative of the concrete computational demands experienced by practitioners while aligning trees.

Indeed, at an intuitive level, this reduced complexity only stems from the fact that the expected degree of a node in a random, uniformly distributed, ordered rooted tree is asymptotically constant, and has low variance. This complexity is provably robust to the introduction of weights in the tree specification, following a classic result by Drmota [7]. It follows that there is typically a constant number of indexing subforests for non-terminals, leading to $\Theta(n_1 \cdot n_2)$ left-hand sides, the evaluation of which can be performed in constant time.

The choice of the random model may even mislead the unsuspecting reader into drawing erroneous conclusions. Indeed, consider the average case analysis of our tree alignment where a random pair of trees of total length $n = n_1+n_2$ is drawn uniformly. As shown in Appendix A, the average-case complexity of our tree alignment algorithm is in $\Theta(n\sqrt{n})$, while the complexities of sequence alignment algorithms are typically in $\Theta(n^2)$. This seems surprising, since tree alignment generalizes sequence alignment, and any subquadratic algorithm for sequence alignment

would have striking consequences in multiple areas of computer science [1].

However, this apparent breakthrough is again without foundations, and only reveals a dissymetry in the size distributions of random uniform pairs of sequences and trees, for a total size n. Indeed, the size of the smallest sequence in a pair of overall size n follows a uniform distribution, while the size of the smallest tree in a pair is in $o(n)$. One should thus exercise caution in the choice of the model, and in the interpretation of an average-case complexity.

Bibliography

[1] A. Abboud, V. V. Williams and O. Weimann, *Consequences of Faster Alignment of Sequences*, *Automata, Languages, and Programming: 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I* eds. J. Esparza, P. Fraigniaud, T. Husfeldt and E. Koutsoupias (Springer Berlin Heidelberg, Berlin, Heidelberg, 2014), Berlin, Heidelberg, pp. 39–51.

[2] H. Andrade, I. Area, J. J. Nieto and A. Torres, The number of reduced alignments between two dna sequences., *BMC Bioinformatics* **15** (2014) p. 94.

[3] G. Blin, A. Denise, S. Dulucq, C. Herrbach and H. Touzet, Alignments of RNA structures, *IEEE/ACM Trans. Comput. Biology Bioinform.* **7**(2) (2010) 309–322.

[4] A. Denise, Y. Ponty and M. Termier, Controlled non-uniform random generation of decomposable structures, *Theoretical Computer Science* **411**(40-42) (2010) 3527–3552.

[5] C. Do, S. Gross and S. Batzoglou, Contralign: Discriminative training for protein sequence alignment, *Research in Computational Molecular Biology*, eds. A. Apostolico, C. Guerra, S. Istrail, P. Pevzner and M. Waterman, *Lecture Notes in Computer Science* **3909** (Springer Berlin Heidelberg, 2006), pp. 160–174.

[6] A. Dress, B. Morgenstern and J. Stoye, The number of standard and of effective multiple alignments, *Applied Mathematics Letters* **11**(4) (1998) 43 – 49.

[7] M. Drmota, Systems of functional equations, *Random Structures and Algorithms* **10**(1-2) (1997) 103–124.

[8] P. Duchon, P. Flajolet, G. Louchard and G. Schaeffer, Boltzmann samplers for the random generation of combinatorial structures, *Combinatorics, Probablity, and Computing* **13**(4–5) (2004) 577–625, Special issue on Analysis of Algorithms.

[9] P. Flajolet, P. Zimmermann and B. Van Cutsem, Calculus for the random generation of labelled combinatorial structures, *Theoretical Computer Science* **132** (1994) 1–35, A preliminary version is available in INRIA Research Report RR-1830.

[10] P. Flajolet, É. Fusy and C. Pivoteau, Boltzmann sampling of unlabelled structures, *Proceedings of the Meeting on Analytic Algorithmics and Combinatorics*, Society for Industrial and Applied Mathematics (2007), pp. 201–211.

[11] P. Flajolet and R. Sedgewick, *Analytic combinatorics* (Cambridge University Press, Cambridge, 2009).

[12] R. Giegerich and H. Touzet, Modeling dynamic programming problems over sequences and trees with inverse coupled rewrite systems, *Algorithms* **7**(1) (2014) 62–144.

[13] C. Herrbach, A. Denise and S. Dulucq, Average complexity of the Jiang-Wang-Zhang pairwise tree alignment algorithm and of a RNA secondary structure alignment algorithm, *Theor. Comput. Sci.* **411**(26-28) (2010) 2423–2432.

[14] M. Höchsmann, T. Töller, R. Giegerich and S. Kurtz, Local similarity in rna secondary structures., *Proc IEEE Comput Soc Bioinform Conf* **2** (2003) 159–168.

[15] M. Höchsmann, B. Voss and R. Giegerich, Pure multiple rna secondary structure align-

ments: A progressive profile approach, *IEEE/ACM Trans. Comput. Biol. Bioinformatics* **1** (January 2004) 53–62.

[16] T. Jiang, L. Wang and K. Zhang, Alignment of trees - an alternative to tree edit, *Theor. Comput. Sci.* **143**(1) (1995) 137–148.

[17] S. Needleman and C. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, *J. Mol. Bio.* **48** (1970) 443–453.

[18] Y. Ponty and C. Saule, A combinatorial framework for designing (pseudoknotted) RNA algorithms, *Algorithms in Bioinformatics - 11th International Workshop, WABI 2011, Saarbrücken, Germany, September 5-7, 2011. Proceedings*, eds. T. M. Przytycka and M. Sagot *Lecture Notes in Computer Science* **6833**, (Springer, 2011), pp. 250–269.

[19] G. Sauthoff, S. Janssen and R. Giegerich, Bellman's gap: A declarative language for dynamic programming, *Proceedings of the 13th International ACM SIGPLAN Symposium on Principles and Practices of Declarative Programming*, *PPDP '11*, (ACM, New York, NY, USA, 2011), pp. 29–40.

[20] S. Schirmer and R. Giegerich, Forest alignment with affine gaps and anchors, applied in RNA structure comparison, *Theor. Comput. Sci.* **483** (2013) 51–67.

[21] A. Torres, A. Cabada and J. J. Nieto, An exact formula for the number of alignments between two dna sequences., *DNA Seq* **14** (Dec 2003) 427–430.

[22] M. Vingron and P. Argos, Determination of reliable regions in protein sequence alignments, *Protein Engineering* **3**(7) (1990) 565–569.

[23] M. S. Waterman, *Introduction to Computational Biology: Maps, Sequences, and Genomes* (CRC Press, 1995).

[24] H. S. Wilf, A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects, *Advances in Mathematics* **24** (1977) 281–291.

Appendix A. Analysis of sequence/tree alignment in the overall size

So far, we have kept the sizes n_1 and n_2 of the two input trees separate in our analysis. However, complexity theory traditionally considers the overall size of the input as being the relevant parameter for complexity analysis. This justifies the average-case analysis as a function of the overall size $n = n_1 + n_2$. For the sake of simplicity, we will restrict our analysis to the alignment of sequences and trees using a single label.

A.1. *Alignment of unary sequences*

Sequence alignments are ubiquitous in Bioinformatics, and can be computed exactly in quadratic time and space using the Needlemann-Wunsch algorithm [17]. This algorithm aligns two sequences a and b uses dynamic programming based on the following decomposition of the search space:

$$\mathsf{NW}(i,j) = \bigcup \begin{cases} \{(a_i,-)\} \times \mathsf{NW}(i-1,j) & \text{if } i > 0 \\ \{(-,b_j)\} \times \mathsf{NW}(i,j-1) & \text{if } j > 0 \\ \{\mathrm{Match}(a_i,b_j)\} \times \mathsf{NW}(i-1,j-1) & \text{if } i,j > 0 \\ \varnothing & \text{if } i = j = 0 \end{cases} \tag{A.1}$$

The complexity of this DP scheme is exactly $\Theta(n_1 \times n_2)$.

Proposition 1. *Needlemann-Wunsch runs in $\Theta(n^2)$ expected time.*

Proof. Consider the random variable F_n that denotes the complexity of running the Needlemann-Wunsch algorithm on a random uniform pair of sequences having overall size n. One clearly has

$$\mathbb{E}(F_n) = \sum_{\substack{w,w' \text{ s.t.} \\ |w|+|w'|=n}} \frac{|w| \times |w'|}{s_n} = \sum_{\substack{n_1,n_2 \text{ s.t.} \\ n_1+n_2=n}} \frac{n_1 \times n_2}{s_n}$$

where $s_n \equiv n+1$ denotes the total number of pairs of (unary) sequences of cumulated length n. Now consider the generating function of sequences

$$\mathsf{Seq}(z) = \frac{1}{1-z} = \sum_{n \geq 0} z^n,$$

one has

$$z \cdot \mathsf{Seq}'(z) = \frac{z}{(1-z)^2} = \sum_{n \geq 0} n \cdot z^n.$$

It follows that

$$\big(z \cdot \mathsf{Seq}'(z)\big)^2 = \frac{z^2}{(1-z)^4} = \left(\sum_{n_1 \geq 0} n_1 \cdot z^{n_1}\right) \times \left(\sum_{n_2 \geq 0} n_2 \cdot z^{n_2}\right)$$
$$= \sum_{n \geq 0} \sum_{n_1+n_2=n} (n_1 \times n_2) \cdot z^n.$$

and consequently

$$\mathbb{E}(F_n) = \frac{[z^n]\, \frac{z^2}{(1-z)^4}}{n+1} \in \Theta(n^2) \qquad \square$$

A.2. *Alignment of unary trees*

Proposition 2. *Our tree alignment algorithm runs in $\Theta(n\sqrt{n})$ expected time.*

Proof. First let us observe that the dominating term in the complexity of tree alignment algorithms is contributed by $\mathcal{V}$, and is due to the investigation of all the possible decompositions for the subforests provided as arguments.

Let G_n be the random variable that denotes the sum of degrees of all the infix/suffix[c] subforests in a pair of trees of overall size n, taken uniformly at random. Our goal is then to compute

$$\mathbb{E}(G_n) = \sum_{\substack{(b,b') \in \mathcal{B}^2 \text{ s.t.} \\ |b|+|b'|=n}} \frac{1}{p_n} \sum_{\substack{(f,f') \in \\ \mathcal{F}_i(b) \times \mathcal{F}_s(b')}} |f| + |f'|$$

where $\mathcal{B}$ denotes the set of ordered trees, $p_n = \sum_{i \geq 0}^{n} |\mathcal{B}_i| \times |\mathcal{B}_{n-i}|$ is the number of pair of trees having cumulated size n, and $\mathcal{F}_i(b)$ (resp. $\mathcal{F}_s(b)$) denotes the set of all infix (resp. suffix) subforests in b.

Let $R(z,u)$ (resp. $S(z,u)$) be the bivariate generating function counting the number of *infix* (resp. *suffix*) forests found in ordered trees, according to both the size of their supporting tree (z), and their degree (u). One has

$$R(z,u) = \sum_{b \in \mathcal{B}} \sum_{f \in \mathcal{F}_i(b)} u^{|f|}\, z^{|b|} \qquad \text{and} \qquad S(z,u) = \sum_{b \in \mathcal{B}} \sum_{f \in \mathcal{F}_s(b)} u^{|f|}\, z^{|b|}.$$

Observe that a specification for $\mathcal{R}$ (resp. $\mathcal{S}$) the set of all infix (resp. suffix) subforests in all trees can be adapted from the classic specification of trees, by first identifying which node should see its children marked, mark some of them, and

[c] An *infix* (sub)forest is any contiguous sequence of siblings of a given node. A (sub)forest is called *suffix* if it is infix, and constitutes a suffix of the siblings of a node.

then proceed with the generation of the remaining subtrees. One obtains

$$\mathcal{R} = \mathcal{Z} \times \mathsf{Seq}(\mathcal{B}) \times \mathcal{R} \times \mathsf{Seq}(\mathcal{B}) + \mathcal{Z} \times \mathcal{R}' \qquad \mathcal{R}' = \mathsf{Seq}(\mathcal{B}) \times \mathsf{Seq}(\mathcal{U} \times \mathcal{B}) \times \mathsf{Seq}(\mathcal{B})$$
$$\mathcal{S} = \mathcal{Z} \times \mathsf{Seq}(\mathcal{B}) \times \mathcal{S} \times \mathsf{Seq}(\mathcal{B}) + \mathcal{Z} \times \mathcal{S}' \qquad \mathcal{S}' = \mathsf{Seq}(\mathcal{B}) \times \mathsf{Seq}(\mathcal{U} \times \mathcal{B})$$
$$\mathcal{B} = \mathcal{Z} \times \mathsf{Seq}(\mathcal{B}) \qquad (\mathsf{Seq}(\mathcal{Y}) := \mathsf{Seq}(\mathcal{Y}) \times \mathcal{Y} + \mathsf{Eps})$$

where $\mathcal{R}'/\mathcal{S}'$ are the trees with a marked (in/suf)fix forest directly below the root.

Solving the associated functional equation gives closed-form equations for $R(z,u)$ and $S(z,u)$. The cumulated size of forests in $\mathcal{R} \times \mathcal{S}$ can then be obtained through an evaluation at $u=1$ of the partial derivative in u

$$\left.\frac{\partial(R(z,u) \times S(z,u))}{\partial u}\right|_{u=1} = \sum_{(b,b') \in \mathcal{B}^2} \sum_{\substack{(f,f') \in \\ \mathcal{F}_i(b) \times \mathcal{F}_s(b')}} (|f| + |f'|)\, z^{|b|+|b'|}$$

and dividing by the total number of pairs of trees gives the expectation of G_n:

$$\frac{[z^n]\left(\left.\frac{\partial(R(z,u) \times S(z,u))}{\partial u}\right|_{u=1}\right)}{[z^n]B(z) \times B(z)} = \sum_{\substack{(b,b') \in \mathcal{B}^2 \text{ s.t.} \\ |b|+|b'|=n}} \frac{1}{p_n} \sum_{\substack{(f,f') \in \\ \mathcal{F}_i(b) \times \mathcal{F}_s(b')}} |f| + |f'| \equiv \mathbb{E}(G_n).$$

We are now left to consider the asymptotics of the two terms appearing in the above expression. Firstly, one has

$$\left.\frac{\partial(R(z,u) \times S(z,u))}{\partial u}\right|_{u=1} = \frac{1}{(1-4z)} \cdot g(z)$$
$$g(z) := \frac{1 - 5z + 5z^2 - (1 - 3z + z^2)\sqrt{1-4z}}{z^2}$$

Clearly the dominant singularity is a pole at $\rho = 1/4$ where $g(\rho) = 1$, and one has

$$[z^n]\left.\frac{\partial(R(z,u) \times S(z,u))}{\partial u}\right|_{u=1} = 4^n \left(1 + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)\right).$$

As for the denominator, one has

$$B(z) \times B(z) = \frac{1 - 2z - \sqrt{1-4z}}{2}$$

and basic singularity analysis yields

$$[z^n]B(z) \times B(z) = \frac{1}{4\sqrt{\pi}} \cdot \frac{4^n}{n\sqrt{n}} \left(1 + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)\right)$$

from which we conclude that

$$\mathbb{E}(G_n) = \frac{4^n \left(1 + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)\right)}{\frac{1}{4\sqrt{\pi}} \frac{4^n}{n\sqrt{n}} \left(1 + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)\right)} = 4\sqrt{\pi} \cdot n\sqrt{n} \left(1 + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)\right). \qquad \square$$