Evolutionary models of structured RNA

Robert K. Bradley[1], Ian Holmes[1,2,*]

1 Biophysics Graduate Group, University of California, Berkeley, CA, USA

2 Department of Bioengineering, University of California, Berkeley, CA, USA

∗ E-mail: indiegram@postbox.biowiki.org

Abstract

The existence of an all-RNA primordial ribosome was proposed by Francis Crick in 1968. Recently, a domain-level model for the original ribosome was proposed by Smith et al. Given the abundance of ribosomal RNA sequence data and the many successful reconstructions of ancestral proteins, the reconstruction and synthesis of ancestral ribosomes (and other RNAs) is a feasible goal for paleogenetics. This will require new bioinformatics tools and methods, in particular a robust theoretical framework for reconstructing histories of substitutions, indels and changes in RNA structure.

We describe a "transducer composition" algorithm for extending pairwise statistical models of RNA structural evolution to models of multiple sequences, related by a phylogenetic tree. This algorithm draws on formal models of computational linguistics as well as the 1985 "protosequence" algorithm of David Sankoff. The output of the composition algorithm is a multiple-sequence stochastic context-free grammar. We describe dynamic programming algorithms, which are robust to null cycles and empty bifurcations, for parsing this grammar. Example applications include structural alignment of non-coding RNAs, propagation of structural information from an experimentally-characterized sequence to its homologs, and inference of the ancestral structure of a set of diverged RNAs.

We implemented these algorithms for a three-taxon phylogeny. Using our implementation on a simple model of pairwise RNA structural evolution, we simulated data under the model and tested the dependence of alignment inference and reconstruction accuracy on the evolutionary distance of the outgroup of the phylogeny. Our results suggest that we may be able to accurately reconstruct ancestral structures despite sequence and structural divergence if the evolutionary process is well-understood. We discuss the implications of our results for RNA algorithm design and phylogenetic reconstruction of RNA evolutionary history.

The programs described are available as part of the DART software package for sequence analysis, released under the GNU Public License at `http://biowiki.org/dart`.

Author Summary

Introduction

In 1968, Francis Crick hypothesized that the first ribosome consisted entirely of RNA, without any protein cofactors [1]. A domain structure for this primeval ribosome was recently proposed [2]. To synthesize such a reconstructed ribosome, or reconstructions of other evolutionarily significant RNAs such as group II introns [3] or telomerase [4], it will be necessary to develop methods that can predict the sequences of ancient RNAs based on the divergent sequences of their many descendants.

Inspection of RNA alignments, for example in the RFAM database [5], suggests that an evolutionary model for RNA structure must eventually include multiple layers of detail: point substitutions, covariant substitutions of base-pairs [6, 7], indels [8], local changes in secondary structure such as helix slippage [9], and changes in domain structure [2]. Stochastic context-free grammars (SCFGs), which can efficiently model the long-range correlations generated by RNA secondary structures, are natural probabilistic models of such phenomena and have been used for ncRNA homology detection [10–13], gene prediction [14, 15], folding [16, 17] and alignment [18, 19, 33].

By analogy with substitution processes, which are well-understood [20], we may split the ancestral reconstruction problem into two halves. The first half is the development of a **pairwise model**, describing the probability distribution $P(Y|X)$ of a descendant (Y) conditional on its immediate ancestor (X). In substitution processes, the pairwise model is a conditional substitution matrix. Often (but not always) the pairwise model is derived from an instantaneous model of change, i.e., a continuous-time Markov chain (parametrized by a rate matrix); obtaining the transition probabilities of this chain (via exponentiation of the rate matrix) yields a pairwise model whose parameters are smoothly-varying functions of a finite time parameter T. A pairwise model represents an individual branch of a phylogenetic tree, with T representing the length of that branch.

The second half of the problem involves extending the model (and related inference algorithms) from a single branch to a complete phylogeny, i.e., from a pairwise model of two sequences to a **multiple-sequence model** of many sequences. In a typical situation, the sequences at the leaves of the tree are observed, but those at internal nodes are unobserved. Questions of interest then include:

A. What is the likelihood for the observed sequence data?

B. Can we sample from (or exactly compute) the posterior distribution of the unobserved sequence at the root node?

C. Can we sample from the posterior of the unobserved sequences at the other internal nodes?

D. Can we estimate summaries of the evolutionary history, such as the number of substitution events on each branch (for a substitution model), the alignment (for a model which includes indels), or changes in the underlying structure (for a model of RNA structure)?

For substitution models, there has been extensive work focused on answering each of these questions. Given a pairwise substitution model, questions A and B can be answered exactly by Felsenstein's pruning algorithm [21] and question C can be answered by the peeling algorithm (first presented for pedigree analysis by Elston and Stewart [22]). The estimation of evolutionary histories (question D) has been addressed by exact summarization [23] and sampling [24] approaches. Many of these questions and answers can be unified by an appropriate framework; for example, another representation of answers A-C is that the "pruning" & "peeling" algorithms (combined) are just the sum-product algorithm on a directed graphical model [25], yielding exact marginal distributions for unobserved variables. Graphical models also suggest general-purpose sampling approaches in addition to the exact sum-product algorithm.

The two halves of the reconstruction problem — developing a pairwise model and then extending it to multiple sequences — are largely independent. Felsenstein's pruning algorithm, for example, is essentially blind to the parametric form of the pairwise substitution model; it just assumes that a substitution model is provided for every branch. Subsequent models developed by other researchers can be plugged into the pruning algorithm without modification [26, 27].

We therefore addressed the problem of modeling the indel-evolution of multiple structured RNAs in a similarly-modular fashion by separating the creation of pairwise and multiple-sequence models. In previous work, we addressed the first (pairwise) part of the RNA reconstruction problem by describing a simple continuous-time model of RNA structural evolution [28]. This model corresponded to a Pair SCFG with a time-dependent parametrization which we used to simultaneously align and predict the structure of related RNAs. The focus of the present work is to solve the second (multiple-sequence) part of the RNA reconstruction problem by giving a general procedure for extending a pairwise model to multiple sequences related by a phylogenetic tree. This process yields a multiple-sequence SCFG, a natural model of the evolutionary relationships between multiple structured RNAs.

The main contributions of this paper are (1) an algorithm that transforms a phylogenetic ensemble of pair grammars into a coherent, multiple-sequence SCFG, (2) dynamic programming (DP) algorithms for

performing inference under this multiple-sequence SCFG, and (3) freely-available software implementing algorithms (1) and (2) for the simplified case of a three-branch star-topology tree. While the idea of composing conditionally-normalized models on trees is intuitive, the resulting models can be very complex, even for simple models of RNA evolution, making (1) necessary. Studies of related indel models have suggested that an implementation of dynamic programming (DP) algorithms on the three-branch tree topology is sufficient to draw samples from the posterior distribution of ancestral sequences on more complex tree topologies, using Markov Chain Monte Carlo or MCMC [29–31], suggesting that (2) and (3) are, in principle, sufficient for analyzing trees relating many sequences.

We show that our algorithm produces a multiple-sequence grammar which is much more compact than suggested by naive approaches to model construction. We provide analyses of the asymptotic complexities of models constructed using our procedure and provide estimates of the time and memory required to reconstruct the structures of several RNA families for the case of a 3-taxon phylogeny, which we have implemented in the program INDIEGRAM. While by these estimates only the smallest sequences currently fit into affordable memory, thereby preventing us from applying our method to many problems of interest, we assess the conduct a simulation study to investigate the accuracy with which we can hope to reconstruct ancestral structures over long evolutionary time in the presence of structural divergence.

In the Discussion, we speculate on algorithmic extensions that may reduce memory requirements, inspired by related work in reconstructing DNA and protein sequences.

Methods

We describe below a general method for constructing a multiple-sequence stochastic grammar for alignment, folding and ancestral reconstruction of RNA, given a phylogenetic tree and a description of the evolutionary process acting along each branch.

Overview

Our problem statement is this: **Given a phylogenetic tree relating several structured RNAs and a description of the evolution of a structured RNA along a single branch of the tree (in the form of a Pair SCFG), (1) find the corresponding phylogenetic multiple-sequence grammar and (2) use that grammar to reconstruct, *a posteriori*, the evolutionary histories of the RNAs.** We assume that the phylogeny, including both the tree topology and branch lengths, is known.

This paper focuses on model construction and inference algorithms rather than the heuristics and constraints which will be necessary to make these algorithms fast enough for analysis of many biological datasets. As discussed below, the complexity of our inference algorithms is prohibitively high for many problems of interest. However, this complexity can be significantly reduced by incorporating outside knowledge. For example, if we know the consensus structure of several sequences or their individual structures, then we can constrain our algorithms accordingly. Similarly, we might consider only ancestral structures which are compatible with a curated multiple alignment. Such constraints are commonly used by programs for RNA sequence analysis [18, 35].

The following sections introduce more precise definitions for two-sequence models of RNA structure and proceed to outline our algorithms for (1) combining these two-sequence models on a phylogenetic tree and (2) using the composite phylogenetic grammars for inference.

Two-sequence models

We discuss the general problem of creating state-space models of the evolution of related sequences, beginning with models of substitution processes acting at independent sites (as studied in likelihood phylogenetics) and generalizing to models of indels first in primary sequences, then in sequences with conserved secondary structure.

A stochastic model for the evolution of one sequence (the ancestor, X) into another (the descendant,

Y) over an interval of time (T) can be described by a joint distribution, $P(X, Y|T)$. This joint distribution can be factored as $P(X) \cdot P(Y|X, T)$, where $P(X)$ is the marginal distribution over ancestral sequences and $P(Y|X, T)$ is the conditional distribution over descendant sequences given an ancestral sequence. In terms of phylogenetics, the conditional distribution $P(Y|X, T)$ describes the evolution $X \stackrel{T}{\rightarrow} Y$ along a branch of length T.

It is possible to "multiply" two such models together. More precisely, one multiplies two conditional distributions and sums out the intermediate sequence. Thus, successive evolution along two branches $X \stackrel{T_1}{\rightarrow} Y \stackrel{T_2}{\rightarrow} Z$ is modeled by the distribution

$$P(Y, Z|X, T_1, T_2) = P(Y|X, T_1)P(Z|Y, T_2)$$

and we can sum sequence Y out of this, obtaining the distribution

$$P(Z|X, T_1 + T_2) = \sum_Y P(Y|X, T_1)P(Z|Y, T_2)$$

for the composite branch $X \stackrel{T_1+T_2}{\longrightarrow} Z$.

This formalism underlies likelihood phylogenetics. Working under the independent-sites assumption, $P(X, Y|T)$ is the (X, Y)'th element of the joint substitution matrix for a single site and $P(Y|X, T)$ is the corresponding element of the conditional matrix. The conditional matrix is in fact the matrix exponential $\exp(\mathbf{R}T)$, where $\mathbf{R}$ is the substitution rate matrix [23]. Composition of two branches just amounts to a matrix multiplication.

A similar formalism can be used to describe the evolution of whole sequences with indels. Suppose that the joint distribution $P(X, Y|T)$ is the distribution modeled by a pair hidden Markov model (Pair HMM) [11], a probabilistic model of the evolution of two sequences under the approximation that only adjacent characters are directly correlated, and the marginal $P(X)$ is the distribution of a single-sequence HMM, a probabilistic model of single sequences under the same approximation. The conditional distribution $P(Y|X, T)$ then corresponds to a conditional Pair HMM: a finite-state machine which transforms one sequence (the input, X) into another (the output, Y). Following computational linguists, we call this conditionally-normalized state machine a **string transducer** or simply a **transducer** [36]. Because of its conditional normalization, this state machine is distinct from a standard Pair HMM. A Pair HMM

has two outputs X and Y and emits symbols to both of those outputs, while a transducer absorbs symbols from the input X and emits symbols to the output Y. Despite this distinction, Pair HMMs and transducers share very similar inference algorithms; for example, $P(Y|X,T)$ is computed using a direct analogue of the Forward algorithm [11].

We extend this formalism to the case of structured RNA as follows. Let X and Y now represent structured RNA sequences, or more precisely, parse trees. A single-sequence SCFG models the marginal $P(X)$; the joint distribution $P(X,Y|T)$ is modeled by a jointly-normalized Pair SCFG [11]. The conditional distribution $P(Y|X,T)$ is modeled by a conditionally-normalized Pair SCFG. Following terminology from computational linguistics [37], we call this conditionally-normalized grammar a **parse-tree transducer**.

String transducers are special cases of parse-tree transducers, just as HMMs are special cases of SCFGs. Henceforth, we will drop the distinction between strings and parse trees. We will also refer interchangeably to "states" (in the state-machine representation) and "nonterminals" (in the grammar representation). Likewise, we will refer interchangeably to "state paths" (machines) and "parse trees" (grammars).

Terminology and normalization. The stochastic state machine which generates parse-trees from the marginal distribution $P(X)$ is a **singlet transducer**. The machine that absorbs parse-trees X and generates modified parse-trees Y from the conditional distribution $P(Y|X,T)$ is a **branch transducer**. Singlet transducers only emit symbols to their output sequence; branch transducers, in contrast, can both emit symbols to their output and absorb symbols from their input sequence.

Much as the states of Pair HMMs with no null states, other than the `Start` and `End` states, can be classified as `Insert` or `Match`, transducers can have states of type `Start`, `End`, `Wait`, `Insert` and `Match`. The first three state types, `Start`, `End` and `Wait`, are null: They do not emit or absorb any symbols and are required solely for "organizational" purposes (see following section). Two types of states can emit and/or absorb symbols, `Insert` and `Match`. An `Insert` state emits a symbol to the output, without absorbing anything. A `Match` state absorbs a symbol on the input and either emits the same symbol to the output, substitutes a different output symbol, or emits no output symbol at all (the latter corresponds to a deletion).

As stated above, the Pair SCFG must be "conditionally normalized" so that models can be chained together, extending the pairwise model to multiple sequences. The transformation rules are partitioned

into co-normalized groups: within each group, the rule probabilities must sum to one. In a jointly-normalized Pair SCFG, each group corresponds to the set of all rules that can be applied to a given nonterminal (i.e., all outgoing transitions from a particular state). In a conditionally-normalized Pair SCFG, the groups are finer-grained: each co-normalized group includes all rules that can be applied to a given nonterminal, *for a given set of absorbed symbols*.

Multiple-sequence models

We can use the concepts of factoring probability distributions introduced in the two-sequence framework to model the common descent of many homologous sequences. Given a phylogenetic tree and two-sequence model, we wish to obtain a multiple-sequence SCFG describing the common descent of the observed sequences.

A singlet transducer (which emits, but does not absorb, symbols) lies at the root of the phylogeny and serves as a generative model of the ancestral sequence. To represent the evolution of an ancestral sequence into many descendant sequences, we place a branch transducer on each branch of the phylogeny. Figures 1 and 2 show simple examples of building a model to represent such an evolution.

Throughout this paper we make frequently refer to two and three-taxon (star) phylogenies. In all cases, the sequence W is assumed to be the (unobserved) ancestral sequence and the sequences X, Y, and Z the (observed) extant sequences (as in Figures 1 and 2).

The composition algorithm

While this composition of conditionally-normalized models on a phylogenetic tree is intuitive, in practice building such an ensemble model is challenging due to the sheer number of possible states and transitions of the ensemble model. The maximum possible state space of the ensemble is the Cartesian product of the individual transducer state spaces. If the singlet transducer has a states, each branch transducer has b states, and there are N branches in the phylogeny, then an upper bound on the number of ensemble states is $O(a \cdot b^N)$. However, in practice there are many fewer states than suggested by this bound; many state configurations are not reachable. For example, for the tree with two extant sequences and a single parent, the branch transducers above leaves X and Y cannot simultaneously be in `Insert` states, as this would correspond to aligning non-homologous (inserted) characters. Similarly, while an upper bound on the number of possible transitions in the transition matrix of the ensemble model is $O((a \cdot b^N)^2)$, in

practice models never reach this bound, due both to inaccessible configurations (such as the one described above) and the sparseness of transitions between the remaining, accessible configurations.

While the accessible state space of the ensemble is smaller than that given by the exponential upper bound, it is generally nonetheless too complex to deal with by hand. For example, the simple model of RNA structural evolution described in Results yields an ensemble model of three sequences with 230 states and 1,789 transitions. More realistic models of RNA give rise to even larger ensemble models.

We therefore need an algorithm to efficiently construct the "state graph" of the ensemble model, consisting of a list of accessible states and the possible transitions between them. By analogy with algorithms for uninformed graph search in artificial intelligence, the transition graph of the ensemble can be constructed by an uninformed depth-first search, where at each step of the search we obtain the next possible ensemble states by changing the state of one or more of the singlet or branch transducers. Beginning with the entire ensemble in state `Start`, the depth-first search of states continues until all nodes are in state `End`.

The allowed transitions of the ensemble can be categorized as follows:

1. **Null Transition**: A branch transducer makes a transition into a `Wait` state, with no terminal emission or bifurcation.

2. **Terminal Emission**: A singlet or branch transducer makes a transition into a state of type `Insert`, emitting left and/or right terminal symbols (e.g., a single base or base-pair). These symbols are absorbed by the immediately-descended transducers, which are pushed into states of type `Match` and may themselves emit terminal symbols that will be absorbed by *their* descendant transducers. This continues down the tree: The terminal symbols are passed from parents, to children, to grandchildren (albeit possibly being replaced by other terminal symbols as they're propagated down) and they propel branch transducers into `Match` states as they go. Eventually, the cascade of emitted terminal symbols stops when all the symbols have been deleted, or when the cascade reaches the leaves of the tree.

3. **Bifurcation**: A singlet or branch transducer makes a transition into a state of type `Insert` that spawns left and/or right nonterminal states. These nonterminals are processed recursively down the tree, just as in a terminal emission (conceptually, a bifurcation is a "nonterminal emission"). As with terminal emissions, absorption of nonterminal emissions propel descendant transducers into

`Match` states, making transitions which may themselves propagate nonterminals further down the tree. A biologically-relevant example of a bifurcation is the insertion of a stem into an ancestral RNA structure, which then may be conserved or deleted in the descendant structures.

4. **End Transition**: The singlet transducer at the root makes a transition to the `End` state, pushing all the descendant branch transducers into `End` states and terminating the current branch of the parse tree.

Co-ordination between the various branch machines is achieved by specifying an ordering on the nodes (preorder with respect to the phylogenetic tree) and by having branch transducers "pause" in `Wait` states while waiting to absorb a symbol from the node above. Only one transducer is allowed to make a spontaneous transition at a time. If this transition corresponds to a terminal emission or a bifurcation, then this may force descendant transducers into making reactive transitions.

The four types of allowed transitions listed above can be formalized simply. Let the total order on the nodes correspond to a preorder traversal of the tree, from which it follows that $m \prec n$ if m is ancestral to n. Let $\mathcal{T}_m$ denote the singlet or branch transducer which emits symbols to node m. Transducer $\mathcal{T}_m$ changes state if and only if one of the following three mutually-exclusive conditions holds:

Type 1: Transducer $\mathcal{T}_m$ is not in a `Wait` state, while all its successor transducers $\mathcal{T}_n$ are in `Wait` states (where $m \prec n$). $\mathcal{T}_m$ is free to make any transition.

Type 2: Transducer $\mathcal{T}_m$ is in a `Wait` state. Its parent transducer emits a symbol, forcing $\mathcal{T}_m$ into a `Match` state so it can absorb that symbol.

Type 3: Transducer $\mathcal{T}_m$ is in a `Wait` state. Its parent transducer enters the `End` state, forcing $\mathcal{T}_m$ into the `End` state as well.

These possible transitions of the ensemble generate multiple alignments as follows:

1. The singlet transducer and all branch transducers begin in their respective `Start` states.

2. Before any residues can appear at the root, the branch transducers all "wind back" into `Wait` states, via type-1 transitions. This occurs in reverse order (i.e.,a postorder traversal of the tree).

3. During this initial windback, clade-specific insertions can occur. This process is described in detail at step 9.

4. With all the branch transducers wound back into `Wait` states, the singlet transducer makes a (type-1) transition into an `Insert` state, emitting a symbol to the sequence at the root node.

5. The transducers on outgoing branches from the root then make (type-2) transitions into `Match` states, either copying the root symbol to their own outputs, substituting it for a different symbol or staying silent (this silence corresponds to a clade-specific deletion; in our formalism, both substitutions and deletions are handled by `Match` states.)

6. The transducers on branches one step away from the root then process the symbols which reached them (if any did), followed by transducers on branches two steps away from the root, then three steps, and so on (these can all be regarded as occurring simultaneously, in a single "wave" or "cascade" of emissions).

7. Eventually the emitted symbols are propagated, via type-2 transitions, all the way to the tips of the tree (if they survived) or to the nodes where they were deleted (if they did not survive). The wave of type-2 transitions has left a lot of branch transducers in `Insert` and `Match` states.

8. The branch transducers then, in postorder, each wind back into `Wait` states, just as at step 2. (These windback transitions can be collapsed into a single ensemble transition, as with the emission cascade; however, the windback may be interrupted by clade-specific insertions; see below.)

9. During the postorder windback, each branch transducer gets an opportunity to generate a new symbol (via type-1 transitions to `Insert` states). (If such a transition to `Insert` occurs, it corresponds to a clade-specific insertion. This insertion is propagated down the tree via a wave of type-2 transitions, as above, then we go back to step 7.)

10. Eventually, the entire ensemble has wound back, so that every transducer is in a `Wait` state except the singlet transducer at the root, which is still in an `Insert` state. At this point, all clade-specific insertions have been processed.

11. The singlet transducer now makes another type-1 transition. If this transition is to an `Insert` state, the entire cycle begins again: the singlet transducer emits the next symbol at the root, and we go back to step 4.

12. If, on the other hand, the singlet transducer enters its `End` state, then a wave of type-3 transitions drives all the branch transducers into their respective `End` states too, bringing the entire ensemble to a halt.

The total sizes of both the state space and the transition matrix are, in general, dramatically smaller than implied by the exponential upper bounds of $O(a \cdot b^N)$ and $O((a \cdot b^N)^2)$. While we do not have provable bounds on the size of the state space, we have observed that the size of the state space is roughly linear in the number of branches, $O(a \cdot b \cdot N)$, and the number of transitions is approximately linear in the number of states, for several pairwise models, including the pairwise model we use here. However, these empirical observations are based on a limited class of pairwise models and we do not have theoretical results for how they will generalize to other pairwise models. We do believe, however, that the worst-case exponential bound will be avoided by (1) omitting inaccessible state configurations and (2) eliminating "windback" states as described in the following section (which we believe will prevent affine gap penalties from generating exponential growth in the number of states).

Therefore, for the models which we have characterized, the search algorithm given below for enumerating all allowed transitions of the ensemble model typically generates $O(b)$ transitions from any given state, thereby creating a very sparse transition matrix of size $O(a \cdot b^2 \cdot N)$.

Inference algorithms for multiple-sequence models

In this section, we describe dynamic programming (DP) algorithms for inferring the alignment, structure and evolutionary history of multiple related RNAs, using the multiple-sequence SCFG we have derived. The model construction algorithm described above creates an evolutionary model which generates a joint distribution over both ancestral and extant sequences. If we regard the ancestral sequences as hidden variables to be marginalized over, then this model corresponds to a jointly-normalized multiple-sequence SCFG (modeling the extant sequences). We can then use the Cocke-Younger-Kasami (CYK) algorithm to impute the maximum-likelihood alignment, secondary structure and common ancestor of the extant sequences, or the Inside algorithm to sample from the joint posterior distribution of these unknowns.

The possibility of fragments of ancestral sequence that are unobserved in any of the descendant sequences, which can be unbounded in length, leads to complications when performing inference on the multiple-sequence SCFG. The SCFG can contain null cycles, corresponding to generation of these unobserved fragments, the existence of which complicates the inference algorithms.

Due to this possibility of unobserved ancestral sequence, the standard CYK and Inside algorithms fail on the ensemble model. Residues in the ancestor can be absent in all extant sequences; the potential number of such unobserved ancestral residues is unbounded. This manifests as cycles of `Null` states within the multiple-sequence SCFG, states which emit to the ancestral sequence but not to any extant sequence. The presence of bifurcations with one or more possibly-empty child branches, corresponding to entire substructures in the ancestral sequence that are absent in any extant sequence, further complicates inference. While there exists a straightforward fix for maximum-likelihood inference with the CYK algorithm, the Inside algorithm requires further computational machinery.

In order to correctly perform maximum-likelihood inference, we only require that the lowest-probability parse corresponds to the parse returned by the CYK algorithm; the entire probability distribution over parses need not be preserved. Because a state path which includes a cycle of `Null` states necessarily has lower probability than an equivalent one without, we can simply ignore such `Null` cycles during CYK. Bifurcations with empty child branches cannot be similarly ignored, but this problem can be addressed by creating equivalent effective direct transitions between the parents of these bifurcation states and their non-empty children. Once these transformations have been performed, maximum-likelihood inference can be performed using a standard three-sequence CYK algorithm on the ensemble model. (Similarly, the null cycle elimination discussed in computational linguistics requires only that the CYK parse be preserved [38].) However, posterior-decoding inference, which uses the Inside and Outside algorithms, required a modified DP algorithm which preserves the entire probability distribution over parses; the transformations just described to make CYK work properly are insufficient for the Inside and Outside algorithms.

There are two possible solutions to this difficulty: We can either create new versions of the standard inference algorithms for SCFGs which can cope with SCFGs with null cycles, or we can modify the structure of the SCFGs generated by our model-construction algorithm such that they are free of null cycles. The first approach of creating new algorithms is simple in principle—we can simply iteratively re-fill entries of the DP matrix, corresponding to summing the probabilities for unobserved null cycles, until the approximation is sufficiently good—but it is difficult to implement efficiently, and necessarily has worse time complexity than the standard inference algorithms due to the need to iterate over parts of the (large) DP matrix. The second approach, of modifying the structure of the SCFGs themselves, has a closed-form solution for the case of HMMs, but to our knowledge, efficient exact null-cycle elimination for SCFGs is

an open mathematical problem (complicated by the presence of null cycles involving bifurcations).

We therefore developed an iterative approximation to remove null cycles from the ensemble model in order to use the standard inference algorithms for SCFGs without incurring the further computational complexity required by modifying the DP algorithms themselves.

Eliminating null cycles in SCFGs

Null cycles in SCFGs constitute a "missing data" problem, wherein the unobserved values of variables contribute to probabilities of observed events. Such missing data problems are common in probabilistic models of sequences and can frequently be dealt with by exactly marginalizing over the probabilities of the unobserved variables. For example, eliminating missing "empty" columns from an independent-sites model of sequence, wherein alignment columns are independent, is straightforward [39] and missing emissions in a HMM can be dealt with by matrix inversion [34, 40, 41] or the "wing retraction" techniques used in HMMER [42]. Algorithms for exact elimination of HMM null cycles by matrix inversion are well-known [34] and are implemented by programs such as PhyloComposer [40] and HMMoC [41]. These algorithms effectively replace the HMM with an equivalent HMM that contains no null cycles but generates the same probability distribution over sequences. However, these solutions do not easily generalize to SCFGs, due to the possible presence of null bifurcations.

Null cycles can be removed from SCFGs by (i) separating bifurcations into those which have one or more empty children (and can therefore be represented using transition or termination rules) and those that have two nonempty children and then (ii) encapsulating all paths that do not emit any terminal symbols (and may therefore cause cyclic dependencies in the DP matrix) as effective direct transitions, with probabilities obtained by geometric series summation of the grammar's transition matrix.

Both (i) empty bifurcations and (ii) paths without terminal symbols generically appear in the ensemble model. An empty bifurcation occurs when a child branch of a bifurcation state transitions to the `End` state without emitting any symbols and can be removed from the model by creating an effective direct transition encapsulating the empty bifurcation. For example, we can create an effective direct transition $N_1 \rightarrow N_3$ between null states N_1 and N_3 in place of the empty bifurcation $N_1 \rightarrow B \rightarrow (N_2\ N_3) \rightarrow (End\ N_3)$, where B is a bifurcation state with children $(N_2\ N_3)$. Bifurcation states are the most computationally-costly part of our models, so it is important to eliminate as many as possible without reducing model expressiveness.

Similarly, our model construction procedure generically produces many "windback" null transitions to non-emitting placeholder states. If `Emit` states E_1 and E_2 are connected in the state graph by `Null` states N_1 and N_2, then the transition $E_1 \rightarrow N_1 \rightarrow N_2 \rightarrow E_2$ with probability $P(E_1 \rightarrow N_1) \cdot P(N_1 \rightarrow N_2) \cdot P(N_2 \rightarrow E_2)$ can be replaced with a transition $E_1 \rightarrow E_2$ with an identical probability.

We give a formal description of the null-cycle elimination procedure, along with procedures for probabilistically "restoring" null cycles to sampled parse trees and Inside-Outside expectation counts, below.

Definitions. Let $G = (\Omega, \mathcal{N}, \mathbf{R}, P)$ be a stochastic context-free grammar (SCFG) consisting of a set of terminal symbols Ω, a set of nonterminal symbols (a.k.a. "states") $\mathcal{N}$, a set $\mathbf{R}$ of *production rules* $L \rightarrow R$ (where $L \in \mathcal{N}$ and $R \in (\mathcal{N} \cup \Omega)^*$) and a non-negative weight function on the rules, $P : (\mathcal{N} \times (\mathcal{N} \cup \Omega)^*) \rightarrow [0, \infty)$.

(The rule set $\mathbf{R}$ can be considered redundant to the specification of the grammar, in that $\rho \notin \mathbf{R} \Leftrightarrow P(\rho) = 0$. The interpretation is that P is a probabilistic weight function and $\mathbf{R}$ is informally the set of rules with non-zero weight.)

Let $\mathcal{L}(A)$ be the set of rules that can be applied to nonterminal $A \in \mathcal{N}$ (i.e., rules in which A appears on the left), and let $\mathcal{R}(B)$ be the set of rules that can generate nonterminal A (i.e.,rules in which A appears on the right). Let $\mathcal{A}(B) = \mathcal{L}(A) \cup \mathcal{R}(A)$ be the set of all rules involving A. Define these also on sets of nonterminals, e.g., $\mathcal{L}(\mathbf{N}) = \bigcup_{A \in \mathbf{N}} \mathcal{L}(A)$ for $\mathbf{N} \subseteq \mathcal{N}$.

Let $\mathcal{T}$ be the set of all parse trees for G. Suppose that parse tree $T \in \mathcal{T}$ makes $n_T(\rho)$ uses of rule ρ; then define the parse tree weight $P(T) = \prod_{\rho \in \mathbf{R}} [P(\rho)]^{n_T(\rho)}$.

If $\sum_{T \in \mathcal{T}} P(T) = 1$, we say that G is *probabilistically normalized by parse tree*. If $\sum_{\rho \in \mathcal{L}(A)} P(\rho) = 1$ for all $A \in \mathcal{N}$, we say that G is *probabilistically normalized by production rule*. Note that normalization by production rule $\Rightarrow$ normalization by parse tree.

Let $S \in \Omega^*$ denote a terminal sequence. Let $\text{seq} : \mathcal{T} \rightarrow \Omega^*$ be the function mapping a parse tree to its terminal sequence. and let $\text{root} : \mathcal{T} \rightarrow \mathcal{N}$ be the function returning the root nonterminal of a parse tree. The *inside probability of S rooted at A* is $P(S|A) = \sum_{T:\text{seq}(T)=S,\text{root}(T)=A} P(T)$. Of particular relevance to null state elimination are the probabilities $P(\epsilon|A)$, where ϵ is the empty sequence. These are the probabilities that a nonterminal will expire without generating any sequence.

Following earlier formalisms [11, 18, 43], we say that the grammar G has *RNA normal-form rules* if each rule in $\mathbf{R}$ takes one of the following four forms:

Termination rules with one nonterminal on the left and the empty string on the right

$$A \rightarrow \epsilon$$

Transition rules with one nonterminal on the left and the right

$$A \rightarrow B$$

Bifurcation rules with one nonterminal on the left and two on the right

$$A \rightarrow B\ C$$

Emission rules with one nonterminal on the left and right and at least one terminal on the right

$$A \rightarrow x\ B$$
$$A \rightarrow B\ y$$
$$A \rightarrow x\ B\ y$$

Here $A, B, C \in \mathcal{N}$ and $x, y \in \Omega$.

Furthermore, we say that the grammar G has *RNA normal-form states* if each nonterminal (state) $A \in \mathcal{N}$ takes one of the following forms:

Null states : $\mathcal{L}(A)$ contains only transition and termination rules.

Bifurcation states : $\mathcal{L}(A)$ contains exactly one bifurcation rule, $A \rightarrow X\ Y$, where $X \neq Y$ and X, Y are both null states.

Emit states : $\mathcal{L}(A)$ contains only emission rules. Further, $\mathcal{L}(A) = \mathcal{R}(B)$ for some null state B. This null state B is called A's *post-emit* state.

Define a *null cycle* to be a nonterminal A and a sequence of rules $\rho_1, \rho_2 \dots \rho_k$ that, when applied consecutively to A, leave A unchanged; that is, $\rho_k(\rho_{k-1} \dots \rho_2(\rho_1(A))) = A$. Define a *null subtree* to be a null cycle using at least one bifurcation rule.

Suppose that $G = (\Omega, \mathcal{N}, \mathbf{R}, P)$ and $G' = (\Omega, \mathcal{N}', \mathbf{R}', P')$ are two grammars. We say that G and G' are *equivalent in sequence* if there is a mapping between nonterminals, $f : \mathcal{N} \to \mathcal{N}'$, such that $P(S|A) = P(S|f(A))$. We say that G and G' are *equivalent in parse* if there is a mapping between parse trees, $g : \mathcal{T} \to \mathcal{T}'$, such that $P'(T') = \sum_{T:T'=g(T)} P(T)$ and we can define

$$P(T|T') = P(T|g(T) = T') = \frac{P(T)}{P'(T')}$$

Note that equivalence in parse $\Rightarrow$ equivalence in sequence.

Suppose that a grammar G contains null cycles. We seek to transform G into a grammar G'' that is equivalent in parse and sequence, but has no null cycles. If G is normalized by parse tree, then G'' will be too; but G'' is not (necessarily) normalized by production rule.

We do the transformation in two steps, $G \to G' \to G''$. We also provide a stochastic "null cycle restoration" algorithm for sampling from $P(T|T'')$.

Eliminating null bifurcations. Note first that any SCFG can be transformed into an equivalent one with RNA normal-form states by adding null states. Without loss of generality, we therefore consider SCFGs with RNA normal-form states.

Let $G = (\Omega, \mathcal{N}, \mathbf{R}, P)$ be such an SCFG. Let $\mathbf{N} \subseteq \mathcal{N}$ be the set of null states in $\mathcal{N}$, **excluding** post-emit states. Let $\mathbf{E} \subset \mathcal{N}$ be the set of post-emit states. Let $\mathbf{B} \subseteq \mathcal{N}$ be the set of bifurcation states.

We here define the grammar G',

$$G' = (\Omega, \mathcal{N}', \mathbf{R}', P')$$

which is equivalent to G in sequence but has no null subtrees.

The new states $\mathcal{N}'$ are defined as follows. Start with $\mathcal{N}$; remove bifurcation states; then introduce a new state N' for every null $N \in \mathbf{N}$ and three new states B^0, B', B^2 for every bifurcation $B \in \mathbf{B}$.

$$\mathcal{N}' = \left(\bigcup_{N \in \mathbf{N}} \{N'\} \right) \cup \left(\bigcup_{B \in \mathbf{B}} \{B^0, B', B^2\} \right) \cup \mathcal{N} \setminus \mathbf{B}$$

The idea is that N' is a null state that is equivalent to null state N for non-empty sequence. That is, for any terminal sequence S,

$$P'(S|N') = \begin{cases} P(S|N) & \text{if } S \neq \epsilon \\ 0 & \text{if } S = \epsilon \end{cases}$$

Similarly, B' is a null state that is equivalent to bifurcation state B for non-empty sequence, while B^2 is a null state that is is equivalent to B for parse trees where both children of B are non-empty.

In contrast, B^0 is **exactly** equivalent to B in sequence. However, null subtrees in B^0 are explicitly accounted for, their probabilities factored into the transitions $B' \rightarrow L'$ and $B' \rightarrow R'$ and the termination $B^0 \rightarrow \epsilon$, using the inside probabilities for empty sequences, $P(\epsilon|L)$ and $P(\epsilon|R)$.

The probabilities $P(\epsilon|X)$ are related by the following system of equations

$$P(\epsilon|X) = P(X \rightarrow \epsilon) + \sum_Y P(X \rightarrow Y)P(\epsilon|Y) + \sum_{L,R} P(X \rightarrow LR)P(\epsilon|L)P(\epsilon|R)$$

which are nonlinearly coupled but may be solved numerically, e.g., by the Newton-Raphson method, or by iterated approximation starting from a lower bound $P(\epsilon|X) \geq 0$.

The new rules and their probabilities are

$$\begin{aligned}
\forall \rho \in (\mathbf{R} \setminus \mathcal{A}(\mathbf{B})): \quad P'(\rho) &= P(\rho) \\
\forall N \in \mathbf{N}: \quad P'(N' \to \epsilon) &= 0 \\
\ldots \forall B \in \mathbf{B}: \quad P'(N' \to B') &= P(N \to B) \\
\ldots \forall M \in \mathbf{N}: \quad P'(N' \to M') &= P(N \to M) \\
\ldots \forall E \in \mathbf{E}: \quad P'(N' \to E) &= P(N \to E) \\
\forall (A \to B) \in \mathcal{R}(\mathbf{B}): \quad P'(A \to B^0) &= P(A \to B) \\
\forall (B \to LR) \in \mathcal{L}(\mathbf{B}): \quad P'(B^0 \to \epsilon) &= P(B \to LR)\ P(\epsilon|L)\ P(\epsilon|R) \\
P'(B^0 \to B') &= 1 \\
P'(B' \to \epsilon) &= 0 \\
P'(B' \to L') &= P(B \to LR)\ P(\epsilon|R) \\
P'(B' \to R') &= P(B \to LR)\ P(\epsilon|L) \\
P'(B' \to B^2) &= 1 \\
P'(B^2 \to L'\ R') &= P(B \to LR)
\end{aligned}$$

Note that grammar G', like G, has RNA normal-form states. Note also that G' is not, in general, normalized by production rule; however, G' is equivalent to G in parse and is therefore normalized by parse tree (if G is).

Eliminating null states. We now proceed to eliminate null cycles from G'. Since null subtrees have been eliminated, the remaining null cycles use only transition rules.

We create the grammar

$$G'' = (\Omega, \mathcal{N}', \mathbf{R}'', P'')$$

which has the same nonterminals as G', but different rules and rule probabilities.

Let $\mathbf{N}' \subseteq \mathcal{N}'$ be the set of null states in $\mathcal{N}'$, **including** post-emit states.

Define $\mathbf{t}$, the *transition matrix* of G', as $t_{XY} = P'(X \to Y)$ for all $X, Y \in \mathcal{N}'$. The effective transition probability q_{XY} between two states X, Y sums over all paths through null states (in the summand, n is the length of the null path):

$$\mathbf{q} = \sum_{n=0}^{\infty} \mathbf{t}^n = (\mathbf{1} - \mathbf{t})^{-1}$$

Here **1** is the identity matrix. The matrix inverse may be computed in the usual ways (Gauss-Jordan elimination, LU decomposition, etc.)

Since the bifurcation states of G' explicitly generate non-empty sequence, the system of equations relating the probabilities $P'(\epsilon|X)$ is completely linear and may be solved in the same way. Let $u_X = P'(\epsilon|X)$ and $v_X = P'(X \to \epsilon)$. Then

$$\mathbf{u} = \mathbf{v} + \mathbf{tu}$$

whose solution is $\mathbf{u} = \mathbf{qv}$. Thus $P'(\epsilon|X) = \sum_Y q_{XY} P'(Y \to \epsilon)$.

We now define P'' as follows.

- For all rules $\rho \in \mathbf{R}'$ that are **not** transitions or terminations, set $P''(\rho) = P'(\rho)$.
- For terminations from null states $X \in \mathbf{N}'$, set $P''(X \to \epsilon) = u_X$. (NB this may create some terminations $X \to \epsilon$ that were not present in G'.)
- For transitions from null states $X \in \mathbf{N}'$ to non-null states $Y \notin \mathbf{N}'$, set $P''(X \to Y) = q_{XY}$. (NB this may create some transitions $X \to Y$ that were not present in G'.)
- For transitions **to** null states $X \in \mathbf{N}'$, set $P''(A \to X) = 0$.

Although we have left null states in G'', they now have no incoming transitions and are inaccessible unless they are post-bifurcation states (i.e.,states which appear on the right-hand side of bifurcation rules), post-emit states, or the start (root) state. All other null states can therefore be dropped from $\mathcal{N}''$.

Restoring null states. Suppose that ρ'' is a rule in G''. Algorithm 1 samples from the distribution of equivalent parse subtrees in G' (possibly containing null cycles). In order to sample from $P(T'|T'')$ we need simply map Algorithm 1 to each rule in T''.

For parameter estimation by Expectation Maximization and some other applications, it is useful to know the expected number of times that a transition was used, summed over the posterior distribution of parse trees (including those with null cycles). If $n''(\rho'')$ is the expected number of times that transition ρ'' was used according to an Inside-Outside computation on G'', then the corresponding expectations $n'(\rho')$ are given by

$$
\begin{aligned}
n'(X \to Y) &= n''(X \to Y)\frac{P''(X \to Y)}{q_{XY}} + \sum_Z n''(X \to Z)\frac{P''(X \to Y)q_{YZ}}{q_{XZ}} \\
n'(X \to \epsilon) &= n''(X \to \epsilon)\frac{P''(X \to \epsilon)}{P''(\epsilon|X)} + \sum_W n''(W \to \epsilon)\frac{q_{WX}P''(X \to \epsilon)}{P''(\epsilon|W)}
\end{aligned}
$$

Expectations for other rules (bifurcations and emissions) are the same for G' as G''.

Restoring null bifurcations. Suppose that ρ' is a rule in G'. Algorithm 2 samples from the distribution of equivalent parse subtrees in G (possibly containing null subtrees). This algorithm also calls Algorithm 3, which samples from the distribution of empty subtrees rooted at a particular nonterminal. In order to sample from $P(T|T')$ we need simply map Algorithm 2 to each rule in T'.

If $n'(\rho')$ is the expected number of times that rule ρ' was used by G', then the corresponding expectations $n(\rho)$ are given by

$$
\begin{aligned}
n(B \to L\ R) &= n'(B' \to L') + n'(B' \to R') + n'(B^0 \to L'\ R') + d(B \to L\ R) \\
n(X \to Y) &= n'(X \to Y) + d(X \to Y) \\
n(X \to \epsilon) &= n'(X \to \epsilon) + d(X \to \epsilon)
\end{aligned}
$$

Expectations for emissions are the same for G as G'. In the above expressions $d(\rho)$ is the expected usage of rule ρ by null subtrees:

$$
d(\rho) = \sum_{(B \to LR) \in \mathbf{R}} \left[n'(B' \to L')c_R(\rho) + n'(B' \to R')c_L(\rho) + n'(B^0 \to \epsilon)(c_L(\rho) + c_R(\rho)) \right]
$$

where $c_W(\rho)$ is the expected usage of rule ρ by an empty parse tree rooted at W, given by

$$
\begin{aligned}
c_X(\rho)P(\epsilon|X) &= P(X \to \epsilon)\delta_{\rho=(X \to \epsilon)} + \sum_Y P(X \to Y)P(\epsilon|Y)\left(c_Y(\rho) + \delta_{\rho=(X \to Y)}\right) \\
&\quad + \sum_{L,R} P(X \to LR)P(\epsilon|L)P(\epsilon|R)\left(c_L(\rho) + c_R(\rho) + \delta_{\rho=(X \to LR)}\right)
\end{aligned}
$$

where δ_U is the Kronecker delta (1 if condition U is true, 0 if it is false). Note that in contrast to the system of equations for $P(\epsilon|X)$, this is a linear system of equations of the form $\mathbf{c} = \mathbf{M}\mathbf{c} + \mathbf{k}$, which may be solved by matrix inversion: $\mathbf{c} = (\mathbf{1} - \mathbf{M})^{-1}\mathbf{k}$.

Dynamic programming algorithms for inference

Once we have performed the transformations described above to remove null cycles from the multiple-sequence SCFGs generated by our model-construction algorithm, we can use the standard CYK, Inside and Outside algorithms for multiple-sequence SCFGs given by Sankoff [44].

The asymptotic time and memory complexities of our inference algorithms are essentially the same as for the Sankoff algorithm [44]: the DP algorithms take memory $O(A \cdot L^{2N})$ and time $O(B \cdot L^{3N})$ for N sequences of length L, where A is the number of (accessible) states in the multiple-SCFG and B is the number of bifurcations. Note that A and B are also dependent on N (see "The TKFST model on a three-taxon phylogeny").

Exact inference on a star phylogeny with N extant sequences therefore has complexities $O(A \cdot L^{2N})$ and $O(B \cdot L^{3N})$ in space and time for a multiple-SCFG with A states and B bifurcations. As described earlier, in practice we frequently have expert knowledge about the structures or evolutionary histories of the sequences of interest, such as a curated multiple alignment. We can use these constraints to reduce the accessible volume of the DP matrix [18]. Algorithms 4–10 give the Inside, CYK, Outside and CYK traceback algorithms for a three-taxon star phylogeny, constrained using the "fold envelopes" described below.

Fold Envelopes. We use the fold envelope concept [28,43] to constrain the set of structures which our algorithms consider. A fold envelope $\mathscr{F}^{(X)}$ for a sequence X is a set of coordinate pairs satisfying

$$\mathscr{F}^{(X)} \subset \left\{(i,j) : 0 \le i \le j \le L^{(X)}\right\} ; \quad L^{(X)} = |X| . \tag{1}$$

We consider a subsequence $x_{i+1} \ldots x_j$ only if the corresponding coordinate pair $(i,j) \in \mathscr{F}^{(X)}$. The unconstrained fold envelope has set equality in Equation 1.

We use two orderings for sequences in the fold envelopes. An inside-outside ordering is used for the iteration in the Inside algorithm: Subsequences are ordered such that each successive subsequence contains all previous subsequences in the fold envelope. More precisely, subsequences in $\mathscr{F}^{(X)}$ are sorted in the

same order as coordinate pairs (i, j) are generated by the iteration $\{$ for $i = L^{(X)}$ to 0 $\{$for $j = i$ to $L^{(X)}\}$ $\}$.

The Outside algorithm uses an outside-inside ordering, where subsequences starting at a fixed i are reverse-sorted with respect to the inside-outside ordering described above. Subsequences in $\mathscr{F}^{(X)}$ are then sorted in the same order as coordinate pairs (i, j) are generated by the iteration $\{$ for $i = 0$ to $L^{(X)}$ $\{$for $j = L^{(X)}$ to $i\}$ $\}$. We frequently refer to subsequences by their index in the fold envelope. The m$^{\text{th}}$ subsequence in $\mathscr{F}^{(X)}$ is labeled $m^{(X)}$ and corresponds to the coordinate pair (i_m, j_m). The index of a pair $(i, j) \in \mathscr{F}^{(X)}$ is $n^{(X)}[i, j]$.

In order to take full advantage of the reduction in computational complexity offered by restricting our inference algorithms to subsequences contained in the fold envelopes, we must avoid iterating over unreachable combinations of subsequences (unreachable because they are not permitted by the fold envelope constraints). An efficient implementation relies on iterators over subsequences in the fold envelope which are connected by production rules of the ensemble grammar. Inward and outward emission connections for a sequence X, specifying which subsequence is reachable from a given subsequence $m^{(X)}$ and ensemble state $\boldsymbol{b}$, are defined as

$$c_{in}\left(\boldsymbol{b}; m^{(X)}\right) = n^{(X)}\left[i_m + \Delta_L^{(X)}(\boldsymbol{b}), j_m - \Delta_R^{(X)}(\boldsymbol{b})\right]$$
$$c_{out}\left(\boldsymbol{b}; m^{(X)}\right) = n^{(X)}\left[i_m - \Delta_L^{(X)}(\boldsymbol{b}), j_m + \Delta_R^{(X)}(\boldsymbol{b})\right],$$

where the quantities $\Delta_L^{(X)}(\boldsymbol{b})$ and $\Delta_R^{(X)}(\boldsymbol{b})$ are the lengths of the left and right emissions of the ensemble state b to the sequence X. (Recall that the m$^{\text{th}}$ subsequence in $\mathscr{F}^{(X)}$ is labeled $m^{(X)}$ and corresponds to the coordinate pair (i_m, j_m).) The emission connection is undefined if the corresponding subsequence is not in the fold envelope. Inward, outward-left and outward-right bifurcation connections, specifying which subsequences are connected by bifurcation production rules of the ensemble SCFG, are defined for a subsequence $n^{(X)}$ as

$$b_{in}(n^{(X)}) = \left\{\left(n_L^{(X)}, n_R^{(X)}\right) : i_{n_L^{(X)}} = i_{n^{(X)}},\, j_{n_L^{(X)}} = i_{n_R^{(X)}},\, j_{n_R^{(X)}} = j_{n^{(X)}}\right\} \tag{2}$$
$$b_{out,L}(n^{(X)}) = \left\{\left(n_O^{(X)}, n_L^{(X)}\right) : i_{n_L^{(X)}} = i_{n_O^{(X)}},\, j_{n_L^{(X)}} = i_{n^{(X)}},\, j_{n^{(X)}} = j_{n_O^{(X)}}\right\} \tag{3}$$
$$b_{out,R}(n^{(X)}) = \left\{\left(n_O^{(X)}, n_R^{(X)}\right) : i_{n^{(X)}} = i_{n_O^{(X)}},\, j_{n^{(X)}} = i_{n_R^{(X)}},\, j_{n_R^{(X)}} = j_{n_O^{(X)}}\right\} \tag{4}$$

We generally write out explicit subsequence coordinate pairs (i, j) when their usage will make mathe-

matical formulae clearer and fold-envelope labels $n^{(X)}$ when writing pseudocode.

The Inside algorithm. The Inside algorithm is used to calculate the likelihood of sequences under an ensemble model. It is analogous to the Forward algorithm for HMMs.

The inside probability $\alpha_{\boldsymbol{a}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$ is the summed probability of the triplet of subsequences $\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$ for sequences X, Y, Z. under all paths through the model which are rooted in state $\boldsymbol{a}$. Algorithm 4 gives pseudocode for the fold-envelope version of the Inside algorithm. The subroutines calcTransEmitProb, calcLBifurcProb and calcRBifurcProb used in the Inside algorithm are defined below.

The transition and emission probability calcTransEmitProb$(\boldsymbol{a}; \cdot)$ can be calculated by iterating over ensemble states $\boldsymbol{b}$ which connect the subsequence triplet $\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$ to others in the fold envelopes. Pseudocode for the constrained calculation is given in Algorithm 5.

The left-bifurcation probability for an ensemble state $\boldsymbol{a}$ bifurcating to two ensemble states, calcLBifurcProb $\left(\boldsymbol{a}; n_L^{(X)}, n_R^{(X)}, n_L^{(Y)}, n_R^{(Y)}, n_L^{(Z)}, n_R^{(Z)}\right)$, is

$$\sum_{\boldsymbol{b} \mid \exists\, \boldsymbol{a} \to \boldsymbol{c}\boldsymbol{b}} P(\boldsymbol{a} \to \boldsymbol{c}\,\boldsymbol{b})\, \alpha_{\boldsymbol{c}}\left(n_L^{(X)}, n_L^{(Y)}, n_L^{(Z)}\right)\, \alpha_{\boldsymbol{b}}\left(n_R^{(X)}, n_R^{(Y)}, n_R^{(Z)}\right)$$

and the right-bifurcation probability for an ensemble state $\boldsymbol{a}$ bifurcating to two ensemble states, calcRBifurcProb $\left(\boldsymbol{a}; n_L^{(X)}, n_R^{(X)}, n_L^{(Y)}, n_R^{(Y)}, n_L^{(Z)}, n_R^{(Z)}\right)$, is

$$\sum_{\boldsymbol{b} \mid \exists\, \boldsymbol{a} \to \boldsymbol{b}\boldsymbol{d}} P(\boldsymbol{a} \to \boldsymbol{b}\,\boldsymbol{d})\, \alpha_{\boldsymbol{b}}\left(n_L^{(X)}, n_L^{(Y)}, n_L^{(Z)}\right)\, \alpha_{\boldsymbol{d}}\left(n_R^{(X)}, n_R^{(Y)}, n_R^{(Z)}\right).$$

The boundary condition of the probability of 0-length subsequences is determined by the probability of transitions to `End`. The termination condition is

$$P\left(X, Y, Z\right) = \alpha_{\texttt{Start}}\left(N^{(X)}, N^{(Y)}, N^{(Z)}\right),$$

where `Start` is the unique start state of the ensemble grammar and $N^{(X)}$ is the outermost subsequence for sequence X, etc.

Note that we are assuming that the transformations described in "Eliminating null states" have been performed, such that there are no cycles of `Null` states as well as no empty bifurcations.

The CYK algorithm. The CYK algorithm is used to calculate the probability of the most-likely state path (or parse) capable of generating the input sequences. It is analogous to the Viterbi algorithm for HMMs.

The CYK algorithm can be obtained from the Inside algorithm by replacing sums over paths through the ensemble model with the max operation. The CYK probabilities for indices $\gamma_{\boldsymbol{a}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$ then represent the probability of the most likely path through the model generating the triplet of subsequences $\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$.

The resulting CYK algorithm is shown in Algorithm 6. The subroutine calcTransEmitProb is defined in Algorithm 7. The subroutines calcLBifurcProb and calcRBifurcProb used in the CYK algorithm are defined as

$$\max_{\boldsymbol{b}|\,\exists\,\boldsymbol{a}\rightarrow\boldsymbol{cb}} P(\boldsymbol{a}\rightarrow\boldsymbol{c}\,\boldsymbol{b})\,\gamma_{\boldsymbol{c}}\left(n_L^{(X)}, n_L^{(Y)}, n_L^{(Z)}\right)\,\gamma_{\boldsymbol{b}}\left(n_R^{(X)}, n_R^{(Y)}, n_R^{(Z)}\right)$$

and

$$\max_{\boldsymbol{b}|\,\exists\,\boldsymbol{a}\rightarrow\boldsymbol{bd}} P(\boldsymbol{a}\rightarrow\boldsymbol{b}\,\boldsymbol{d})\,\gamma_{\boldsymbol{b}}\left(n_L^{(X)}, n_L^{(Y)}, n_L^{(Z)}\right)\,\alpha_{\boldsymbol{d}}\left(n_R^{(X)}, n_R^{(Y)}, n_R^{(Z)}\right).$$

The Outside algorithm. The Outside algorithm is primarily used an an intermediary for posterior decoding on the model. It is analogous to the Backward algorithm for HMMs.

The outside probability $\beta_{\boldsymbol{b}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$ for an ensemble state $\boldsymbol{b}$ is the summed probability of the sequences X, Y, Z under all paths through the ensemble model which are rooted in the start state of the model, excluding all paths for the triplet of subsequences $\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$ which are rooted in the ensemble state $\boldsymbol{b}$. Algorithm 8 gives pseudocode for the fold-envelope version of the Outside algorithm. The subroutines calcTransEmitProb, calcLBifurcProb and calcRBifurcProb used in the Outside algorithm are defined below.

As with the Inside and CYK algorithms, the transition and emission probability calcTransEmitProb can be calculated efficiently using the subsequence connections defined earlier (Algorithm 9). The left-bifurcation probability calcLBifurcProb $\left(\boldsymbol{b}; n_O^{(X)}, n_L^{(X)}, n_O^{(Y)}, n_L^{(Y)}, n_O^{(Z)}, n_L^{(Z)}\right)$ is

$$\sum_{\boldsymbol{a}|\,\exists\,\boldsymbol{a}\rightarrow\boldsymbol{cb}} P(\boldsymbol{a}\rightarrow\boldsymbol{c}\,\boldsymbol{b})\,\beta_{\boldsymbol{a}}\left(n_O^{(X)}, n_O^{(Y)}, n_O^{(Z)}\right)\,\alpha_{\boldsymbol{c}}\left(n_L^{(X)}, n_L^{(Y)}, n_L^{(Z)}\right)$$

and the right-bifurcation probability calcRBifurcProb $\left(\boldsymbol{b}; n_O^{(X)}, n_R^{(X)}, n_O^{(Y)}, n_R^{(Y)}, n_O^{(Z)}, n_R^{(Z)}\right)$ is

$$\sum_{\boldsymbol{a} \mid \exists\, \boldsymbol{a} \to \boldsymbol{b}\boldsymbol{d}} P(\boldsymbol{a} \to \boldsymbol{b}\,\boldsymbol{d})\, \beta_{\boldsymbol{a}} \left(n_O^{(X)}, n_O^{(Y)}, n_O^{(Z)}\right)\, \alpha_{\boldsymbol{d}} \left(n_R^{(X)}, n_R^{(Y)}, n_R^{(Z)}\right)$$

The boundary condition is just

$$\beta_{\texttt{Start}} \left(N^{(X)}, N^{(Y)}, N^{(Z)}\right) = 1\,,$$

where $N^{(X)}$ is the outermost subsequence for sequence X, etc.

The CYK traceback algorithm. The CYK traceback algorithm, in combination with the CYK algorithm, is used to find the most-likely state path generating the extant sequences (in other words, the maximum-likelihood parse generating the observed data). It is analogous to the Viterbi traceback algorithm for HMMs. Algorithm 10 gives the constrained CYK traceback algorithm.

Results

Automated grammar construction

We implemented our model construction algorithm on the three-taxon star phylogeny. Given a singlet transducer modeling ancestral structures and a branch transducer modeling structural evolution, our Perl modules generate C++ code for the corresponding jointly-normalized three-sequence (Triplet) SCFG. Any model of structural evolution which can be represented as a Pair SCFG and factored into singlet and branch transducers is permitted as input to the packages, allowing for flexible, automated model design.

A simple model of RNA structural evolution

We illustrated our method for building models of structured sequences using a simplified model which was introduced in previous work, the TKF Structure Tree model [28], a simple probabilistic model of the evolution of RNA structure.

The TKF Structure Tree (TKFST) model is based on the Thorne-Kishino-Felsenstein (TKF) model of the stochastic evolution of primary sequences via indel events [45]. In the original TKF model, sequence evolves under a time-homogeneous linear birth-death-immigration process [46]. Single characters are inserted with rate λ and deleted with rate μ. At equilibrium, sequences obey a geometric length distribution with parameter κ. Although this model has flaws (e.g., it lacks affine gap penalties, rate heterogeneity and context-dependent mutation rates), it illustrates many of the key ideas used by more sophisticated indel models, notably the possibility for systematic derivation of pairwise alignment automata from first principles via analysis of birth-death processes [45, 47].

The TKF Structure Tree model is an extension of the TKF model to RNA structure. In this model, loop and stem regions are mutually nested (Figure 3). Both loops and stems have geometric length distributions with parameters κ_1 and κ_2. Single bases are inserted and deleted in loops with rates λ_1 and μ_1; similarly, base-pairs are inserted and deleted in stems with rates λ_2 and μ_2. Insertions of a new stem into an existing loop sequence (or deletions of an existing stem) occur at the same rate as single-base insertions (or deletions) and can model large-scale structural changes (Figure 4).

We parametrized the singlet and branch transducers of the TKFST model using estimates reported by a phylo-grammar for RNA secondary structure prediction, PFOLD [16], and an implementation of pairwise alignment for the TKF Structure Tree model, EVOLDOER [28]. The equilibrium distributions

of unpaired and paired nucleotides of the singlet and branch transducers, as well as the substitution models of unpaired and paired nucleotides of the branch transducers, were derived from the substitution rate matrices of the PFOLD program. These rate matrices, which have proven useful for RNA structure prediction [16,17,48], were derived from the Bayreuth tRNA database [49] and the European large subunit rRNA database [50].

This continuous-time model corresponds to a Pair SCFG and as such fits neatly into our modeling framework once the probability distribution is appropriately factored into marginal and conditional distributions (generated by singlet and branch transducers). Tables 1 and 2 show the states and transitions of the singlet transducer (single-sequence SCFG) which generates ancestral sequence under the Structure Tree model. Tables 3 and 4 show the states and transitions of the branch transducer (conditionally-normalized Pair SCFG) which evolves a sequence and structure along a branch of the phylogenetic tree.

The equilibrium distribution and transition probabilities between states of the TKFST model can be expressed in terms of functions of the evolutionary time along a branch and the insertion and deletion rates λ_1 and μ_1 of the model. The length of ancestral sequences is geometric in κ_1 (Table 2), defined as

$$\kappa_1 = \lambda_1/\mu_1 \,.$$

The three functions $\alpha(t)$, $\beta(t)$ and $\gamma(t)$ which govern the transition probabilities in Table 4 are defined for loop sequences as

$$\begin{aligned}
\alpha_1(t) &= \exp\left(-\mu_1 t\right) \\
\beta_1(t) &= \frac{\lambda_1 \left(1 - \exp\left((\lambda_1 - \mu_1)t\right)\right)}{\mu_1 - \lambda_1 \exp\left((\lambda_1 - \mu_1)t\right)} \\
\gamma_1(t) &= 1 - \frac{\mu_1 \left(1 - \exp\left((\lambda_1 - \mu_1)t\right)\right)}{\left(1 - \exp(-\mu_1 t)\right)\left(\mu_1 - \lambda_1 \exp\left((\lambda_1 - \mu_1)t\right)\right)}
\end{aligned}$$

and similarly for stem sequences [28].

Assessing TKFST as a model of RNA structure

The TKFST model, like the original TKF model, probably needs refinements in order to be useful. For example, it fails to model certain phenomena observed in natural RNA structures (such as base-stacking or tetraloops) and in alignments of those structures (such as helix slippage). We assessed its

appropriateness as a model of RNA structural evolution by conducting benchmarks of its capabilities for (1) multiple sequence alignment of structured RNAs, summing over all possible structures, and (2) structure prediction of homologous structured RNAs and comparing its performance to one of the best-performing pairwise SCFGs for RNA multiple alignment [33]. The results of these benchmarks, reported in Table 5 and Table 6, suggest that TKFST is a useful guide for deriving more complicated models of RNA evolution: while it has relatively poor sensitivity (but high positive predictive value) as a base-pairing predictor, it is competitive with one of the most accurate RNA multiple sequence alignment programs [33].

TKFST's poorer performance at base-pairing prediction is likely due to its much-simpler model of RNA structure. The richer grammar, as described in [18], is richer than TKFST: excluding the substitution model, it has 14 free parameters (compared to TKFST's 4), uses an affine gap penalty (compared to TKFST's linear gap penalty), and explicitly models structural features such as multiple-branched loops, symmetric/asymmetric bulges, and minimum loop lengths. Unlike TKFST, the richer grammar is structurally unambiguous: a one-to-one mapping exists from structures to parse trees. Although we use the TKFST model as an illustrative example of a Pair SCFG that can be extended with our method, the model is not fundamental to our approach and can be replaced by a different and more realistic pairwise model, such as the pairwise SCFG used in these comparisons [33]. We anticipate that further improvements should be possible by reviewing other comparisons of SCFGs at structure prediction, such as the study by [17].

The TKFST model on a three-taxon phylogeny

We used our model-construction algorithm to build the grammar corresponding to the TKFST model acting on a star phylogeny with three (extant) leaf sequences and a single (unobserved) ancestral sequence. We chose this phylogeny for two reasons: (1) it is the simplest extension of the well-studied, standard two-sequence (Pair SCFG) model and (2) algorithms on a phylogeny with three leaves should be sufficient for ergodic sampling of reconstructions on any larger phylogeny, using an MCMC kernel similar to that of [30].

The statistics of the TKFST model on the three-taxon phylogeny illustrate the advantages of our procedure for model construction. While the singlet and branch transducers are relatively simple—the singlet transducer, shown in Table 2, has 7 total states and 2 bifurcation states and the branch

transducer, shown in Table 4, has 21 total states and 6 bifurcation states—the ensemble model of three extant sequences is very complex. The naive exponential upper-bound gives a maximal state space of size $O(7 \cdot 21^3) \Rightarrow 6 \cdot 10^4$ states. Using our uninformed search algorithm, we determine that there are 287 accessible states and 686 possible transitions between these states (compare with the $287^2 \simeq 8 \cdot 10^4$ transitions estimated with the exponential calculation). After performing the transformations described in "Eliminating null cycles in SCFGs" to eliminate useless windback states, the ensemble model has a reduced state space with 230 states, albeit at the cost of extra transitions, bring the total to 1,789 transitions (here we are trading reduced memory complexity, which is linear in the number of states, for increased time complexity, which is linear in the number of transitions). Note that both before and after the reduction in complexity, the total number of states and transitions are less than the approximate bounds of $O(a \cdot b \cdot N) = O(7 \cdot 21 \cdot 3) \Rightarrow 441$ states and $O(a \cdot b^2 \cdot N) = O(7 \cdot 21^2 \cdot 3) \Rightarrow 9,261$ transitions given in "The composition algorithm."

Text S? contains the complete model specification, including states and transitions, produced by our model composition algorithm. The extreme complexity of the ensemble model, despite the simplicity of the underlying model of RNA structure, makes clear the necessity for automated procedures for model construction.

We implemented constrained maximum-likelihood inference of the structural alignment and ancestral structure of three extant sequences (Algorithms 6-10) in a C++ program (INDIEGRAM). For tractability, INDIEGRAM uses the concept of fold envelopes described earlier to limit the fold space considered by the CYK algorithm, permitting structural information for the three extant sequences to be (optionally) supplied as input. If no structural information is supplied, then INDIEGRAM uses a single-sequence SCFG to estimate a set of plausible folds [18], which are used to constrain the CYK algorithm.

The inference algorithms in INDIEGRAM could be further constrained to enforce, for example, a fixed multiple alignment or consensus structure of extant sequences. While experimentally-determined structures of individual RNAs are rare, curated deep sequence alignments, such as those constructed for ribosomal RNAs [51], are more frequently available. By constraining the inference algorithms with such sequence alignments, the memory and time complexity of the algorithms could be dramatically reduced. Such constraints can be naturally expressed with "alignment envelopes," the sequence-level counterpart of fold envelopes [18]. However, in this paper we focus on model construction and inference algorithms and postpone exploration of heuristics and constraints of these algorithms for future work.

Reconstructing small RNAs with the TKFST model

While reconstructing large RNAs such as ribosomal subunits is currently computationally-inaccessible without further heuristics to constrain our algorithms, reconstructing small RNAs of biological interest will soon be feasible. We used the TKF Structure Tree model on a 3-taxon phylogeny to reconstruct most-probable ancestral structures of three *nanos* 3' translational control elements (TCEs) and three tRNAs, two small RNAs whose structures are well-known and which are accessible given current computational limitations. While the phylogenetic trees relating the tested sequences have short branch lengths, making the reconstruction problem easy by forcing the reconstructed structures to be essentially-identical to those of one of the extant RNAs (reconstructed structures not shown; see `http://biowiki.org/IndieGram`), these are nonetheless useful proof-of-concept datasets.

Table 7 shows estimates of the memory and time required to reconstruct biologically-interesting subunits of the two RNAs tested, the *nanos* 3' translational control element and tRNAs, as well as two small RNAs which show significant structural divergence, the Y RNAs and Group II introns, and therefore promise to be interesting candidates for ancestral reconstruction.

The dependence of reconstruction accuracy on outgroup branch length

Guided by our experience with the *nanos* 3' TCE and tRNA, where the reconstruction problem was made easy by the presence of a close outgroup, we conducted a simulation study to investigate the dependence of reconstruction accuracy on outgroup branch length. We simulated the evolution of RNAs under the TKFST model along 3-taxon phylogenies, where we kept the branch lengths of the two sibling species constant and varied the branch length of the outgroup. We fixed the tree topology as $((X:1, Y:1):1, Z:z)$, where the branch length z of the outgroup was varied between 0.5 and 3.0 and a branch length of 1 corresponds to one expected substitution per site in loop sequence. We chose the parameters of the TKFST model to generate "interesting" alignments (~ 0.3 substructure indels per alignment) and considered only alignments with at least two ancestral stems, sequences of length $\in [30, 70]$, loops of length $\in [3, 10]$ and stems of lengths $\in [1, 20]$.

Figure 5 shows the dependence of alignment accuracy and ancestral reconstruction accuracy on the branch length of the outgroup Z.

While these results correspond to a best-case scenario in which we have a perfectly-parametrized

evolutionary model, recent results demonstrating that inference of sequence alignment is surprisingly robust to mis-parametrized models [41] suggest that the same may be true for inference of ancestral structures.

Discussion

Following the conception of paleogenetics [54], a large number of synthetic reconstructions of ancient proteins have been reported in the literature [55–62]. There are also the beginnings of a movement to reconstruct ancient DNA [63–69]. Given the importance of the RNA world hypothesis to current discussions of the origin of life [70–76], the many modern-day relics of this world [77–80] and the recent proposal of a structural model for the primordial ribosome [2], we believe that phylogenetic reconstruction of ancient RNA is a significant problem, deserving of strong bioinformatics support.

The work reported in this paper builds on substantial prior art in the areas of evolutionary modeling and ancestral reconstruction. The reconstruction of ancient sequences was first proposed in 1963 by Pauling and Zuckerkandl [54]; current applications of this idea, mostly using substitution models, are surveyed in the book edited by Liberles [81]. Many algorithms in phylogenetics implicitly reconstruct substitution histories, whether by parsimony [82, 83] or likelihood [21]. There has been much effort to model the evolution of sequences related by a tree with models permitting indels [29, 34, 36, 45, 84–91]. Recent work has extended these ideas to the reconstruction of indel histories [92, 93], particularly at the genomic scale [69, 94]. There is substantial prior work in computational linguistics on the theory of transducers for sequences [95] and parse trees [37, 38, 96, 97] (from which we take the terms "string transducer" and "parse-tree transducer"). We draw on the bioinformatics literature for SCFGs [10,11,98], especially Pair SCFGs [14, 18, 19] and phylogenetic SCFGs [16]. In particular, an early example of a pairwise conditional model $P(Y|X)$ for structure-dependent RNA evolution may be found in [12]. A conditional framework similar to ours in some respects is described by Sakakibara *et al* [99]. The statistical inference algorithms for multiple-sequence SCFGs are closely related to the "protosequence algorithm" of Sankoff [44], a seminal work in both RNA structural alignment and ancestral reconstruction.

While we have focused on the TKF Structure Tree model in our Results, our model-construction algorithm is applicable to any model of the evolution of secondary structure which can be expressed as a parse-tree transducer. Realistic structural and thermodynamic effects—such as base-stacking or loop length distributions—can, in principle, be incorporated.

An implementation of inference algorithms for models on the three-taxon phylogeny is sufficient to construct an MCMC sampling algorithm over many sequences on an arbitrary phylogeny. A sketch of such a sampling algorithm is as follows: at each step of the sampling algorithm, we re-sample the sequence

and structure of the ancestral node W, conditioned on the sequences and structures of X, Y and Z. The structural alignment of all four sequences can change at each step, providing for fast mixing and guaranteeing ergodicity. This move is similar to the sampler proposed by [30] for models with a HMM structure. Note that this, in principle, permits construction of a crude sampler to simultaneously infer phylogeny as well, by proposing and accepting or rejecting changes to the underlying tree as well as the implied structural alignment.

Reconstructing structural changes of large RNAs using the three-way sampling kernel which we have described would require resources far in excess of what are currently available; barring the availability of supercomputers with terabytes of memory, such algorithms will only be feasible for short RNAs (Table 7). Thus, alternate strategies will have to be explored. One promising direction is to consider variations on the three-way sampling kernel, such as the importance-sampling approach described for the TKF model by [31]. This approach first proposes an ancestor W by aligning extant sequence X to Y (ignoring Z); then, in a second step, the proposed W is independently aligned to Z. The proposed three-way alignment and reconstruction is then randomly accepted (or rejected) using a Hastings ratio based on the three-way transducer composition. The complexity of this kernel is the same as the pairwise case; with suitable constraints, this is feasible for RNA grammars on present hardware, at least for ribosomal domains (if not yet whole subunits—although pairwise alignment of those should also be possible soon). The approach of Redelings and Suchard therefore merits future consideration in the context of modeling the evolution of RNAs on a tree.

The evolutionary models we have described form a theoretical foundation for studying the structural evolution of RNA gene families in detail. By opening the door to ancestral sequence reconstruction and paleogenomics, these algorithms will allow us to test our knowledge of RNA evolution by direct experimental investigation of early ribonucleic machines.

Acknowledgments

We thank Jeff Thorne, Jamie Cate, Jennifer Doudna, Jotun Hein, David Haussler, Sean Eddy, Elena Rivas and Eric Westhof for inspirational discussions. We are grateful to Ben Redelings and one anonymous referee for illuminating criticism.

References

1. Crick FH (1968) The origin of the genetic code. Journal of Molecular Biology 38:367–379.
2. Smith TF, Lee JC, Gutell RR, Hartman H (2008) The origin and evolution of the ribosome. Biology Direct 3:16.
3. Lehmann K, Schmidt U (2003) Group II introns: structure and catalytic versatility of large natural ribozymes. Critical Reviews in Biochemistry and Molecular Biology 38:249–303.
4. Antal M, Boros E, Solymosy F, Kiss T (2002) Analysis of the structure of human telomerase RNA in vivo. Nucleic Acids Research 30:912–920.
5. Griffiths-Jones S, Bateman A, Marshall M, Khanna A, Eddy SR (2003) Rfam: an RNA family database. Nucleic Acids Research 31:439–441.
6. Hancock JM, Tautz D, Dover GA (1988) Evolution of the secondary structures and compensatory mutations of the ribosomal RNAs of Drosophila melanogaster. Molecular Biology and Evolution 5:393–414.
7. Leontis NB, Stombaugh J, Westhof E (2002) Motif prediction in ribosomal RNAs: Lessons and prospects for automated motif prediction in homologous RNA molecules. Biochimie 84:961–973.
8. Yokoyama T, Suzuki T (2008) Ribosomal RNAs are tolerant toward genetic insertions: evolutionary origin of the expansion segments. Nucleic Acids Research 36:3539–3551.
9. Hancock JM, Dover GA (1990) 'compensatory slippage' in the evolution of ribosomal RNA genes. Nucleic Acids Research 18:5949–5954.
10. Eddy SR, Durbin R (1994) RNA sequence analysis using covariance models. Nucleic Acids Research 22:2079–2088.

11. Durbin R, Eddy S, Krogh A, Mitchison G (1998) Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge, UK: Cambridge University Press.

12. Klein R, Eddy SR (2003) Noncoding RNA gene detection using comparative sequence analysis. BMC Bioinformatics 4.

13. Nawrocki E, Eddy S (2007) Query-dependent banding (QDB) for faster RNA similarity searches. PLoS Computational Biology 3:e56.

14. Rivas E, Eddy SR (1999) A dynamic programming algorithm for RNA structure prediction including pseudoknots. Journal of Molecular Biology 285:2053–2068.

15. Pedersen JS, Bejerano G, Siepel A, Rosenbloom K, Lindblad-Toh K, et al. (2006) Identification and classification of conserved RNA secondary structures in the human genome. PLoS Computational Biology 2:e33.

16. Knudsen B, Hein J (2003) Pfold: RNA secondary structure prediction using stochastic context-free grammars. Nucleic Acids Research 31:3423–3428. Evaluation Studies.

17. Dowell RD, Eddy SR (2004) Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. BMC Bioinformatics 5:71.

18. Holmes I (2005) Accelerated probabilistic inference of RNA structure evolution. BMC Bioinformatics 6.

19. Dowell RD, Eddy SR (2006) Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. BMC Bioinformatics 7:400.

20. Felsenstein J (2003) Inferring Phylogenies. Sinauer Associates, Inc. ISBN 0878931775.

21. Felsenstein J (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. Journal of Molecular Evolution 17:368–376.

22. Elston RC, Stewart J (1971) A general model for the genetic analysis of pedigree data. Human Heredity 21:523–542.

23. Holmes I, Rubin GM (2002) An Expectation Maximization algorithm for training hidden substitution models. Journal of Molecular Biology 317:757–768.

24. Nielsen R (2001) Mutations as missing data: inferences on the ages and distributions of nonsynonymous and synonymous mutations. Genetics 159:401–411.

25. J P (1982) Reverend Bayes on inference engines: A distributed hierarchical approach. In: Proceedings American Association of Artificial Intelligence National Conference on AI. Pittsburgh, PA, pp. 133–136.

26. Yang Z (1994) Estimating the pattern of nucleotide substitution. Journal of Molecular Evolution 39:105–111.

27. Whelan S, Goldman N (2001) A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. Molecular Biology and Evolution 18:691–699.

28. Holmes I (2004) A probabilistic model for the evolution of RNA structure. BMC Bioinformatics 5.

29. Holmes I, Bruno WJ (2001) Evolutionary HMMs: a Bayesian approach to multiple alignment. Bioinformatics 17:803–820.

30. Jensen J, Hein J (2002) Gibbs sampler for statistical multiple alignment. Dept. of Theoretical Statistics, University of Aarhus, Aarhus, Denmark. Technical Report No. 429.

31. Redelings BD, Suchard MA (2005) Joint bayesian estimation of alignment and phylogeny. Systematic Biology 54:401–418.

32. Kiryu H, Tabei Y, Kin T, Asai K (2007) Murlet: a practical multiple alignment tool for structural RNA sequences. Bioinformatics 23:1588–1598.

33. Bradley RK, Pachter L, Holmes I (2008) Specific alignment of structured RNA: Stochastic grammars and sequence annealing. Bioinformatics .

34. Holmes I (2003) Using guide trees to construct multiple-sequence evolutionary HMMs. Bioinformatics 19 Suppl. 1:i147–157.

35. Rivas E, Eddy SR (2001) Noncoding RNA gene detection using comparative sequence analysis. BMC Bioinformatics 2.

36. Bradley RK, Holmes I (2007) Transducers: an emerging probabilistic framework for modeling indels on trees. Bioinformatics 23:3258–3262.

37. Comon H, Dauchet M, Gilleron R, Löding C, Jacquemard F, et al. (2007) Tree Automata Techniques and Applications, Available from: `http://www.grappa.univ-lille3.fr/tata`, chapter 6 ("Tree Transducers"). Release October, 12th 2007.

38. Gécseg F, Steinby M (1997) Handbook of formal languages, Volume 3: Beyond Words, Springer-Verlag, chapter Tree languages.

39. Rivas E, Eddy S (2008) Probabilistic phylogenetic inference with insertions and deletions. PLoS Computational Biology 4:e1000172.

40. Holmes I (2007) Phylocomposer and Phylodirector: Analysis and Visualization of Transducer Indel Models. Bioinformatics 23:3263–3264.

41. Lunter G (2007) HMMoC–a compiler for hidden Markov models. Bioinformatics 23:2485–2487.

42. HMMER. `http://hmmer.wustl.edu/`.

43. Holmes I, Rubin GM (2002) Pairwise RNA structure comparison using stochastic context-free grammars. Pacific Symposium on Biocomputing, 2002.

44. Sankoff D (1985) Simultaneous solution of the RNA folding, alignment, and protosequence problems. SIAM Journal of Applied Mathematics 45:810–825.

45. Thorne JL, Kishino H, Felsenstein J (1991) An evolutionary model for maximum likelihood alignment of DNA sequences. Journal of Molecular Evolution 33:114–124.

46. Kendall DG (1948) On the generalized birth-and-death process. Annals of Mathematical Statistics 19:1–15.

47. Feller W (1971) An Introduction to Probability Theory and its Applications, Vol II. New York: John Wiley and Sons.

48. Bradley RK, Uzilov AV, Skinner M, Bendana YR, Varadarajan A, et al. (2008) Non-coding RNA gene predictors in Drosophila. Submitted.

49. Sprinzl M, Horn C, Brown M, Ioudovitch A, Steinberg S (1998) Compilation of tRNA sequences and sequences of tRNA genes. Nucleic Acids Research 26:148–53.

50. Rijk PD, Caers A, Peer YVd, Wachter RD (1998) Database on the structure of large ribosomal subunit RNA. Nucleic Acids Research 26:183–6.

51. Gutell RR (1993) Collection of small subunit (16S and 16S-like) ribosomal RNA structures. Nucleic Acids Research 21:3051–3054.

52. Crucs S, Chatterjee S, Gavis ER (2000) Overlapping but distinct RNA elements control repression and activation of nanos translation. Molecular cell 5:457–467.

53. Hopfield JJ (1978) Origin of the genetic code: a testable hypothesis based on tRNA structure, sequence, and kinetic proofreading. Proceedings of the National Academy of Sciences of the United States of America 75:4334–4338.

54. Pauling L, Zuckerkandl E (1963) Chemical paleogenetics, molecular "restoration studies" of extinct forms of life. Acta Chemica Scandinavica 17:S9–S16.

55. Malcolm BA, Wilson KP, Matthews BW, Kirsch JF, Wilson AC (1990) Ancestral lysozymes reconstructed, neutrality tested, and thermostability linked to hydrocarbon packing. Nature 345:86–89.

56. Stackhouse J, Presnell SR, McGeehan GM, Nambiar KP, Benner SA (1990) The ribonuclease from an extinct bovid ruminant. FEBS letters 262:104–106.

57. Zhang J, Rosenberg HF (2002) Complementary advantageous substitutions in the evolution of an antiviral RNase of higher primates. Proceedings of the National Academy of Sciences of the United States of America 99:5486–5491.

58. Gaucher EA, Thomson JM, Burgan MF, Benner SA (2003) Inferring the palaeoenvironment of ancient bacteria on the basis of resurrected proteins. Nature 425:285–288.

59. Thomson JM, Gaucher EA, Burgan MF, De Kee DW, Li T, et al. (2005) Resurrecting ancestral alcohol dehydrogenases from yeast. Nature Genetics 37:630–635.

60. Chang BSW, Jönsson K, Kazmi MA, Donoghue MJ, Sakmar TP (2002) Recreating a functional ancestral archosaur visual pigment. Molecular biology and evolution 19:1483–1489.

61. Sun H, Merugu S, Gu X, Kang YY, Dickinson DP, et al. (2002) Identification of essential amino acid changes in paired domain evolution using a novel combination of evolutionary analysis and in vitro and in vivo studies. Molecular Biology and Evolution 19:1490–1500.

62. Ortlund EA, Bridgham JT, Redinbo MR, Thornton JW (2007) Crystal structure of an ancient protein: evolution by conformational epistasis. Science 317:1544–8.

63. Ivics Z, Hackett PB, Plasterk RH, Izsvak Z (1997) Molecular reconstruction of Sleeping Beauty, a Tc1-like transposon from fish, and its transposition in human cells. Cell 91:501–10.

64. Adey NB, Tollefsbol TO, Sparks AB, Edgell MH, Hutchison CA (1994) Molecular resurrection of an extinct ancestral promoter for mouse l1. Proceedings of the National Academy of Sciences of the United States of America 91:1569–1573.

65. Blanchette M, Green ED, Miller W, Haussler D (2004) Reconstructing large regions of an ancestral mammalian genome in silico. Genome Res 14:2412–2423. Comparative Study.

66. Noonan JP, Coop G, Kudaravalli S, Smith D, Krause J, et al. (2006) Sequencing and analysis of Neanderthal genomic DNA. Science 314:1113–1118.

67. Gaucher EA (2007) Ancestral sequence reconstruction as a tool to understand natural history and guide synthetic biology: realizing and extending the vision of zuckerkandl and pauling. In: Liberles [81], chapter 2.

68. Elias I, Tuller T (2007) Reconstruction of ancestral genomic sequences using likelihood. Journal of Computational Biology 14:216–37.

69. Paten B, Herrero J, Fitzgerald S, Flicek P, Holmes I, et al. (2008) Genome-wide nucleotide level mammalian ancestor reconstruction. Accepted.

70. Marintchev A, Wagner G (2004) Translation initiation: structures, mechanisms and evolution. Quarterly reviews of biophysics 37:197–284.

71. Muller AWJ (2005) Thermosynthesis as energy source for the RNA world: a model for the bioenergetics of the origin of life. Bio Systems 82:93–102.

72. Cavalier-Smith T (2006) Rooting the tree of life by transition analyses. Biology Direct 1:19.

73. Martin W, Koonin EV (2006) Introns and the origin of nucleus-cytosol compartmentalization. Nature 440:41–45.

74. Forterre P (2006) Three RNA cells for ribosomal lineages and three DNA viruses to replicate their genomes: a hypothesis for the origin of cellular domain. Proceedings of the National Academy of Sciences of the USA 103:3669–3674.

75. Yakhnin AV (2007) A model for the origin of protein synthesis as coreplicational scanning of nascent RNA. Origins of life and evolution of the biosphere : the journal of the International Society for the Study of the Origin of Life 37:523–536.

76. Danchin A, Fang G, Noria S (2007) The extant core bacterial proteome is an archive of the origin of life. Proteomics 7:875–889.

77. Lee RC, Feinbaum RL, Ambros V (1993) The C. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. Cell 75:843–854.

78. Lowe TM, Eddy SR (1999) A computational screen for methylation guide snoRNAs in yeast. Science 283:1168–1171.

79. Eddy SR (2001) Non-coding RNA genes and the modern RNA world. Nature Reviews Genetics 2:919–929.

80. Mandal M, Boese B, Barrick JE, Winkler WC, Breaker RR (2003) Riboswitches control fundamental biochemical pathways in Bacillus subtilis and other bacteria. Cell 113:577–586.

81. Liberles DA, editor (2007) Ancestral Sequence Reconstruction. Oxford University Press.

82. Edwards AWF, Cavalli-Sforza L (1963) The reconstruction of evolution. Annals of Human Genetics 27:105.

83. D HM, D P (1982) Branch and bound algorithms to determine minimal evolutionary trees. Mathematical Biosciences 59:277–290.

84. Hein J, Wiuf C, Knudsen B, Moller MB, Wibling G (2000) Statistical alignment: computational properties, homology testing and goodness-of-fit. Journal of Molecular Biology 302:265–279.

85. Hein J (2001) An algorithm for statistical alignment of sequences related by a binary tree. In: Altman RB, Dunker AK, Hunter L, Lauderdale K, Klein TE, editors, Pacific Symposium on Biocomputing. Singapore: World Scientific, pp. 179–190.

86. Steel M, Hein J (2001) Applying the Thorne-Kishino-Felsenstein model to sequence evolution on a star-shaped tree. Applied Mathematics Letters 14:679–684.

87. Knudsen B, Miyamoto M (2003) Sequence alignments and pair hidden Markov models using evolutionary history. Journal of Molecular Biology 333:453–460.

88. Lunter GA, Miklós I, Song YS, Hein J (2003) An efficient algorithm for statistical multiple alignment on arbitrary phylogenetic trees. Journal of Computational Biology 10:869–889.

89. Miklós I, Lunter G, Holmes I (2004) A long indel model for evolutionary sequence alignment. Molecular Biology and Evolution 21:529–540.

90. Lunter GA, Drummond AJ, Miklós I, Hein J (2004) Statistical alignment: Recent progress, new applications, and challenges. In: Nielsen R, editor, Statistical methods in Molecular Evolution, Springer Verlag, Series in Statistics in Health and Medicine.

91. Satija R, Pachter L, Hein J (2008) Combining statistical alignment and phylogenetic footprinting to detect regulatory elements. Bioinformatics 24:1236–1242.

92. Kim J, Sinha S (2007) Indelign: a probabilistic framework for annotation of insertions and deletions in a multiple alignment. Bioinformatics 23:289–297.

93. Diallo AB, Makarenkov V, Blanchette M (2007) Exact and heuristic algorithms for the indel maximum likelihood problem. Journal of Computational Biology 14:446–461.

94. Ma J, Zhang L, Suh BB, Raney BJ, Brian J, et al. (2006) Reconstructing contiguous regions of an ancestral genome. Genome Research 16:1557–1565. Comparative Study.

95. Mohri M, Pereira F, Riley M (2000) Weighted finite-state transducers in speech recognition. ISCA ITRW Automatic Speech Recognition :97–106.

96. Rounds WC (1970) Mappings and grammars on trees. Mathematical Systems Theory 4:257–287.

97. Thatcher JW (1970) Generalized sequential machine maps. Journal of Computer and System Sciences 4:339–367.

98. Sakakibara Y, Brown M, Hughey R, Mian IS, Sjölander K, et al. (1994) Stochastic context-free grammars for tRNA modeling. Nucleic Acids Research 22:5112–5120.

99. Sakakibara Y (2003) Pair hidden Markov models on tree structures :232–240Evaluation Studies.

100. Thompson JD, Plewniak F, Poch O (1999) A comprehensive comparison of multiple sequence alignment programs. Nucleic Acids Research 27:2682–2690.

101. Gardner PP, Wilm A, Washietl S (2005) A benchmark of multiple sequence alignment programs upon structural RNAs. Nucleic Acids Research 33:2433–2439.

102. Teunissen S, Kruithof M, Farris A, Harley J, Venrooij W, et al. (2000) Conserved features of Y RNAs: a comparison of experimentally derived secondary structures. Nucleic Acids Research 28:610–619.

103. Klosterman PS, Uzilov AV, Bendana YR, Bradley RK, Chao S, et al. (2006) XRate: a fast prototyping, training and annotation tool for phylo-grammars. BMC Bioinformatics 7.

104. Griffiths-Jones S, Moxon S, Marshall M, Khanna A, Eddy SR, et al. (2005) Rfam: annotating non-coding RNAs in complete genomes. Nucleic Acids Research 33:D121–4.

105. Hofacker IL, Fontana W, Stadler PF, Bonhoeffer S, Tacker M, et al. (1994) Fast folding and comparison of RNA secondary structures. Monatshefte für Chemie 125:167–188.

Figure Legends

Figure 1. A simple example of transducer composition to build an evolutionary model of two extant sequences. An ancestral sequence W evolves into two descendant sequences X and Y. A singlet transducer (the horizontal gray box) emit ancestral sequence and structure and two branch transducers (the gray boxes labeled T_X and T_Y) mutate it according to the specified evolutionary model. Gray nodes correspond to observed data and white nodes unobserved data.

Figure 2. Transducer composition on a star phylogeny with three (extant) leaf sequences. An ancestral sequence W evolves into three descendant sequences X, Y and Z. A singlet transducer (the horizontal gray box) emit ancestral sequence and structure and three branch transducers (the gray boxes labeled T_X, T_Y and T_Z) mutate it according to the specified evolutionary model. Gray nodes correspond to observed data and white nodes unobserved data. If the branch transducers are time-reversible, then this star phylogeny with three leaves is the neighborhood of any interior node in a (binary) phylogenetic tree, from which it follows that evaluating the likelihood function on this star phylogeny is sufficient for a sampling algorithm on an arbitrary phylogeny.

Figure 3. The TKF Structure Tree model represents the evolution of RNA structure as nested stem and loop sequences. The model consists of recursively nested loop sequences (gray, horizontal) and stem sequences (black, vertical). The loops are sequences of unpaired bases and the stems are sequences of covarying base-pairs. Both loop and stem sequences evolve according to the Thorne-Kishino-Felsenstein (TKF) model [45] of molecular evolution. Figure is extended from a similar version in [28].

Figure 4. Evolution of a RNA structure under the TKF Structure Tree model. The TKF Structure Tree model includes phenomena such as point mutations in loop sequences ($1 \rightarrow 2$ and $4 \rightarrow 5$), covariant mutations in stem sequences ($2 \rightarrow 3$), insertions in loop sequences ($3 \rightarrow 4$), insertions in stem sequences ($5 \rightarrow 6$), structural insertions ($6 \rightarrow 7$), and structural deletions ($7 \rightarrow 8$). Figure is extended from a similar version in [28].

Figure 5. Dependence of reconstruction accuracy on outgroup branch length. We allowed the outgroup branch length z to vary within $[0.5, 3.0]$, where a branch length of 1 corresponds to one expected substitution per site in loop sequence. We divided this interval into 25 bins and simulated 25 alignments for each bin. Parameters used were:

Tables

Table 1. State types of the singlet transducer (single-sequence SCFG) of the TKF Structure Tree model.

State	type	absorb	emit	description
L	`Start`			Start of a loop
I_L	`Insert`		(x, null)	Single-base emission
S	`Start`			Start of a stem
I_S	`Insert`		(x, y)	Base-pair emission
B	`Insert`		$(L\ S)$	Bifurcation

Singlet transducers can only have states of type `Start` or `Insert`.

Table 2. Singlet transducer (single-sequence SCFG) of the TKF Structure Tree model.

Source	→	Destination	probability	Source	→	Destination	probability
L	→	$u\ I_L$	$\kappa_l \cdot \pi_l(u)$	S	→	$u\ I_S\ v$	$\kappa_s \cdot \pi_s(uv)$
	|	B	$\kappa_l \cdot \pi_l(S)$		|	B_e	$1 - \kappa_s$
	|	`End`	$1 - \kappa_l$				
I_L	→	$u\ I_L$	$\kappa_l \cdot \pi_l(u)$	I_S	→	$u\ I_S\ v$	$\kappa_s \cdot \pi_s(uv)$
	|	B	$\kappa_l \cdot \pi_l(S)$		|	B_e	$1 - \kappa_s$
	|	`End`	$1 - \kappa_l$				
B	→	$(L\ S)$	1				
B_e	→	$(L$ `End`$)$	1				

The state types for this model are shown in Table 1. The singlet transducer generates ancestral RNA sequences and structures. We use the notation of formal grammars to represent state transformation rules; for example, the rule $I_L \to u\ I_L$ corresponds to (in a Pair HMM) an `Insert` state I_L emitting a nucleotide u and then making a self-transition. Both loop (L and I_L) and stem (S and I_S) sequence evolve as TKF sequences with length parameters κ_l and κ_s (defined in "A simple model of RNA structural evolution"). $\pi_l(u)$ and $\pi_s(uv)$ are the equilibrium distributions of unpaired nucleotides x and paired nucleotides (u, v). The bifurcation state B_e is used to end stem sequences (only loop sequences are allowed to transition to the empty string).

Table 3. State types of the branch transducer (conditionally-normalized Pair SCFG) of the TKF Structure Tree model.

State	type	absorb	emit	description
L	`Start`			Start of a loop
I_L	`Insert`		(u, null)	Single-base insertion
M_L	`Match`	(x, null)	(u, null)	Single-base substitution
D_L	`Match`	(x, null)		Single-base deletion
W_L	`Wait`			Wait for next base
S	`Start`			Start of a stem
I_S	`Insert`		(u, v)	Base-pair insertion
M_S	`Match`	(x, y)	(u, v)	Base-pair substitution
D_S	`Match`	(x, y)		Base-pair deletion
W_S	`Wait`			Wait for next base-pair
B_i	`Insert`		$(L_i\, S_i)$	Stem insertion
B	`Match`	$(L\, S)$	$(L\, S)$	Stem conservation
B_p	`Match`	$(L\, S)$	$(L\, \texttt{End})$	Stem deletion
B_e	`Match`	$(L\, \texttt{End})$	$(L\, \texttt{End})$	Stem extinction

States which have the same names as states of the singlet transducer in Table 1 are the branch-transducer equivalents of the corresponding singlet-transducer states (e.g., a `Match` state might be the branch equivalent of an `Insert` state). States L_i and S_i are the `Start` states of a sub-model (not shown) identical in structure to the singlet transducer. They are used to insert a new stem-loop structure.

Table 4. Branch transducer (conditionally-normalized Pair SCFG) of the TKF Structure Tree model.

Source	→	Destination	probability	Source	→	Destination	probability
L	→	$w\ I_L$	$\beta_l(t) \cdot \pi_l(w)$	S	→	$w\ I_S\ x$	$\beta_s(t) \cdot \pi_s(wx)$
	|	B_i	$\beta_l(t) \cdot \pi_l(S)$		|	W_S	$1 - \beta_s(t)$
	|	W_L	$1 - \beta_l(t)$				
I_L	→	$w\ I_L$	$\beta_l(t) \cdot \pi_l(w)$	I_S	→	$w\ I_S\ x$	$\beta_s(t) \cdot \pi_s(wx)$
	|	B_i	$\beta_l(t) \cdot \pi_l(S)$		|	W_S	$1 - \beta_s(t)$
	|	W_L	$1 - \beta_l(t)$				
M_L	→	$w\ I_L$	$\beta_l(t) \cdot \pi_l(w)$	M_S	→	$w\ I_S\ x$	$\beta_s(t) \cdot \pi_s(wx)$
	|	B_i	$\beta_l(t) \cdot \pi_l(S)$		|	W_S	$1 - \beta_s(t)$
	|	W_L	$1 - \beta_l(t)$				
D_L	→	$w\ I_L$	$\gamma_l(t) \cdot \pi_l(w)$	D_S	→	$w\ I_S\ x$	$\gamma_s(t) \cdot \pi_s(wx)$
	|	B_i	$\gamma_l(t) \cdot \pi_l(S)$		|	W_S	$1 - \gamma_s(t)$
	|	W_L	$1 - \gamma_l(t)$				
W_L	→	$u\ w\ M_L$	$\alpha_l(t) \cdot M_l(u, w)$	W_S	→	$u\ w\ M_S\ x\ v$	$\alpha_s(t) \cdot M_s(uv, wx)$
	|	$u\ D_L$	$1 - \alpha_l(t)$		|	$u\ D_S\ v$	$1 - \alpha_s(t)$
	|	B	$\alpha_l(t)$		|	B_e	1
	|	B_p	$1 - \alpha_l(t)$				
	|	`End`	1				
B	→	$L\ S$	1	B_p	→	L `End`	1
B_i	→	$L_i\ S_i$	1	B_e	→	L `End`	1

The state types for this model are shown in Table 3. The branch transducer evolves a sequence and structure along a branch of the phylogenetic tree. States L_i and S_i are the `Start` states for a sub-model corresponding to an insertion of a new stem in the descendant sequences; the sub-model (not shown) is identical in structure to the singlet transducer shown in Table 2. $\pi_l(w)$ and $\pi_s(wx)$ are the equilibrium distributions of, respectively, descendant unpaired nucleotide w and descendant paired nucleotides (w, x); $M_l(u, w)$ and $M_l(uv, wx)$ are the conditional distributions (i.e. match probabilities) of, respectively, descendant unpaired nucleotide w given an ancestral unpaired nucleotide u and descendant paired nucleotides (w, x) given ancestral nucleotides (u, v). The functions $\alpha_n(t)$, $\beta_n(t)$ and $\gamma_n(t)$ are parametrized by the insertion and deletion rates of the TKFST model and are defined in "A simple model of RNA structural evolution."

Table 5. Percentage sensitivity and positive predictive value (Sensitivity / PPV) for pairwise nucleotide-level alignments in the BRalibaseII benchmark.

	U5	g2intron	rRNA	tRNA
TKFST grammar	81.6 / 81.7	**75.4** / **75.0**	91.4 / 92.6	**94.6** / **94.4**
STEMLOC grammar	**82.6** / **83.7**	74.2 / 74.8	**92.6** / **92.8**	93.2 / 93.9

We compared the performance of the TKFST model for progressive multiple alignment of RNAs against the performance of a grammar with a richer model of RNA structure ("STEMLOC"; [18]). Sensitivity is defined as $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$ and PPV is defined as $\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$, where TP is the number of true positives (correctly aligned residue pairs), FN is the number of false negatives (residue pairs that should have been aligned but were not) and FP is the number of false positives (residue pairs that were incorrectly aligned). These statistics are summed over all pairs of sequences in the multiple alignment; therefore, "Sensitivity" for pairwise residue alignments is equivalent to the Sum of Pairs Score, or SPS [100]. "g2intron" is the RFAM entry `Intron_gpII`, containing domains V and VI of the Group II intron.

Table 6. Percentage sensitivity and positive predictive value (Sensitivity / PPV) for predicted base-pairs in the BRalibaseII benchmark.

	U5	g2intron	rRNA	tRNA
TKFST grammar	37.9 / 68.0	42.1 / 63.8	37.4 / 66.5	70.9 / **88.3**
STEMLOC grammar	74.9 / **73.9**	64.3 / 56.7	51.0 / 59.0	74.0 / 76.4

We compared the performance of the TKFST model for structure-prediction accuracy during progressive multiple alignment of RNAs against the performance of a grammar with a richer model of RNA structure ("STEMLOC"; [18]). "g2intron" is the RFAM entry `Intron_gpII`, containing domains V and VI of the Group II intron.

Table 7. Estimates of the memory and time required to reconstruct ancestral structures of three RNAs from several families of biological interest (as reported by INDIEGRAM).

Family	Sequence lengths	Memory	Time
nanos 3' TCE	61-64 nt	3 Gb	3 min
tRNA	69-73 nt	11 Gb	19 min
Y RNA	47-81 nt	33 Gb	70 min
Group II intron (domains V and VI)	76-91 nt	122 Gb	90 min

The *nanos* 3' translational control element (TCE) sequences are the seed sequences of the corresponding RFAM family [5] and the three tRNA sequences are from the BRalibaseII database [101] (identifiers AB042432.1-14140_14072, Z82044.1-16031_16103 and AC008670.6-83725_83795) The Group II intron sequences (identifiers Z00044.1-87253_87177, X57546.1-2817_2907 and X04465.1-2700_2775) are from BralibaseII [101]. The Y RNAs are hY1, hY4, and hY5 from [102]; sequence lengths exclude the conserved stem S1. The time estimates are for a 2.2GHz AMD Opteron 848 CPU.

Algorithms

Input: Rule $\rho'' \in \mathbf{R}''$
Output: Parse (sub)tree $T' \in \mathcal{T}'$
switch ρ'' **do**
  case $X \to Y$
    if `Random[0,1]` $< P(X \to Y)/q_{XY}$ **then**
      return $(X \to Y)$;
    else
      Let $z = \sum_W q_{XW} P'(W \to Y)$;
      Sample state V from probability distribution $P(V) = q_{XV} P'(V \to Y)/z$;
      return (restoreTransitions$(X \to V) \to Y$) ;
    end
  end
  case $X \to \epsilon$
    if `Random[0,1]` $< P'(X \to \epsilon)/P'(\epsilon|X)$ **then**
      return $(X \to \epsilon)$;
    else
      Let $z = \sum_W q_{XW} P'(W \to \epsilon)$;
      Sample state V from probability distribution $P(V) = q_{XV} P'(V \to \epsilon)/z$;
      return (restoreTransitions$(X \to V) \to \epsilon$) ;
    end
  end
  otherwise
    return (ρ'') ;
  end
end

Algorithm 1: Subroutine restoreTransitions(ρ'') for the null cycle elimination procedure.

Input: Rule $\rho' \in \mathbf{R}'$
Output: Parse (sub)tree $T \in \mathcal{T}$
switch ρ' **do**
  case $B^0 \rightarrow \epsilon$
    Let $B \rightarrow L\ R$ be the original bifurcation in $\mathbf{R}$;
    $T_L \leftarrow$ sampleNullSubtree(L) ;
    $T_R \leftarrow$ sampleNullSubtree(R) ;
    return $(B \rightarrow T_L\ T_R)$;
  end
  case $B' \rightarrow L'$
    Let $B \rightarrow L\ R$ be the original bifurcation in $\mathbf{R}$;
    $T_R \leftarrow$ sampleNullSubtree(R) ;
    return $(B \rightarrow L\ T_R)$;
  end
  case $B' \rightarrow R'$
    Let $B \rightarrow L\ R$ be the original bifurcation in $\mathbf{R}$;
    $T_L \leftarrow$ sampleNullSubtree(L) ;
    return $(B \rightarrow T_L\ R)$;
  end
  case $B^2 \rightarrow L'\ R'$
    Let $B \rightarrow L\ R$ be the original bifurcation in $\mathbf{R}$;
    return $(B \rightarrow L\ R)$;
  end
  case $B^0 \rightarrow B'$ **return** () ;
  case $B' \rightarrow B^2$ **return** () ;
  otherwise
    Let ρ be the original rule in $\mathbf{R}$;
    return (ρ) ;
  end
end

Algorithm 2: Subroutine restoreBifurcations(ρ') for the null cycle elimination procedure.

Input: Nonterminal $A \in \mathcal{N}$
Output: Parse tree $T \in \mathcal{T} : \mathrm{seq}(T) = \epsilon, \mathrm{root}(T) = A$
if *A is a bifurcation state, $A \rightarrow X\ Y$,* **then**
  $T_X \leftarrow$ sampleNullSubtree(X) ;
  $T_Y \leftarrow$ sampleNullSubtree(Y) ;
  return $(A \rightarrow T_X\ T_Y)$;
else
  if `Random[0,1]` $< P(A \rightarrow \epsilon)/P(\epsilon|A)$ **then**
    return $(A \rightarrow \epsilon)$;
  else
    Let $z = \sum_Y P(A \rightarrow Y)P(\epsilon|Y)$;
    Sample state X from probability distribution $P(X) = P(A \rightarrow X)P(\epsilon|X)/z$;
    $T_X \leftarrow$ sampleNullSubtree(X) ;
    return $(A \rightarrow T_X)$;
  end
end

Algorithm 3: Subroutine sampleNullSubtree(A) for the null cycle elimination procedure.

Input: sequences X, Y, Z
foreach $n^{(X)} \in \mathscr{F}^{(X)}$ **do** `/* inside-outside sorted */`
  foreach $n^{(Y)} \in \mathscr{F}^{(Y)}$ **do** `/* inside-outside sorted */`
    foreach $n^{(Z)} \in \mathscr{F}^{(Z)}$ **do** `/* inside-outside sorted */`
      foreach *state* $\boldsymbol{a}$ **do**
        bifurcProb $\leftarrow 0$;
        foreach $\left(n_L^{(X)}, n_R^{(X)}\right) \in b_{in}\left(n^{(X)}\right)$ **do**
          foreach $\left(n_L^{(Y)}, n_R^{(Y)}\right) \in b_{in}\left(n^{(Y)}\right)$ **do**
            foreach $\left(n_L^{(Z)}, n_R^{(Z)}\right) \in b_{in}\left(n^{(Z)}\right)$ **do**
              bifurcProb += `calcLBifurcProb` $(\boldsymbol{a}; \cdot)$;
              bifurcProb += `calcRBifurcProb` $(\boldsymbol{a}; \cdot)$;
            end
          end
        end
        $\alpha_{\boldsymbol{a}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right) \leftarrow$ `calcTransEmitProb` $\left(\boldsymbol{a}; n^{(X)}, n^{(Y)}, n^{(Z)}\right)$;
        $\alpha_{\boldsymbol{a}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$ += bifurcProb ;
        store $\alpha_{\boldsymbol{a}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$;
      end
    end
  end
end
return $\alpha_{\boldsymbol{a}}\left(n^{(X)}[0, L^{(X)}], n^{(Y)}[0, L^{(Y)}], n^{(Z)}[0, L^{(Z)}]\right)$;

Algorithm 4: The constrained Inside algorithm for three sequences X, Y, Z. Ensemble states $\boldsymbol{a}$ in the iteration over states are sorted in Inside fill order with `Emit` states first, then `Null` states in reverse topological order.

Input: state $\boldsymbol{a}, n^{(X)}, n^{(Y)}, n^{(Z)}$, intermediate Inside matrix α
emitProb $\leftarrow 0$;
foreach $\boldsymbol{b} : \exists\, \boldsymbol{a} \rightarrow \boldsymbol{b}$ **do**
  emitProb += $P\left(\boldsymbol{a} \rightarrow \boldsymbol{b}\right) \alpha_{\boldsymbol{b}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$;
end
foreach $\boldsymbol{b} : \exists\, \boldsymbol{a} \rightarrow \boldsymbol{l}\, \boldsymbol{b}\, \boldsymbol{r}$ **do**
  if $c_{in}\left(\boldsymbol{b}; n^{(X)}\right) \notin \mathscr{F}^{(X)}$ *or* $c_{in}\left(\boldsymbol{b}; n^{(Y)}\right) \notin \mathscr{F}^{(Y)}$ *or* $c_{in}\left(\boldsymbol{b}; n^{(Z)}\right) \notin \mathscr{F}^{(Z)}$ **then** next;
  emitProb += $P\left(\boldsymbol{a} \rightarrow \boldsymbol{l}\, \boldsymbol{b}\, \boldsymbol{r}\right) \alpha_{\boldsymbol{b}}\left(c_{in}\left(\boldsymbol{b}; n^{(X)}\right), c_{in}\left(\boldsymbol{b}; n^{(Y)}\right), c_{in}\left(\boldsymbol{b}; n^{(Z)}\right)\right)$;
end
return emitProb;

Algorithm 5: Subroutine calcTransEmitProb() for the Inside algorithm. $\boldsymbol{a}$ and $\boldsymbol{b}$ are ensemble states; $\boldsymbol{l}$ and $\boldsymbol{r}$ are left and right terminal emissions.

Input: sequences X, Y, Z
foreach $n^{(X)} \in \mathscr{F}^{(X)}$ **do** `/* inside-outside sorted */`
  foreach $n^{(Y)} \in \mathscr{F}^{(Y)}$ **do** `/* inside-outside sorted */`
    foreach $n^{(Z)} \in \mathscr{F}^{(Z)}$ **do** `/* inside-outside sorted */`
      foreach *state* $\boldsymbol{a}$ **do**
        bifurcProb $\leftarrow 0$;
        foreach $\left(n_L^{(X)}, n_R^{(X)}\right) \in b_{in}\left(n^{(X)}\right)$ **do**
          foreach $\left(n_L^{(Y)}, n_R^{(Y)}\right) \in b_{in}\left(n^{(Y)}\right)$ **do**
            foreach $\left(n_L^{(Z)}, n_R^{(Z)}\right) \in b_{in}\left(n^{(Z)}\right)$ **do**
              bifurcProb += `calcLBifurcProb` $(\boldsymbol{a}; \cdot)$;
              bifurcProb += `calcRBifurcProb` $(\boldsymbol{a}; \cdot)$;
            end
          end
        end
        $\gamma_{\boldsymbol{a}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$
          $\leftarrow \max\left(\texttt{calcTransEmitProb}\left(\boldsymbol{a}; n^{(X)}, n^{(Y)}, n^{(Z)}\right), \text{bifurcProb}\right)$;
        store $\gamma_{\boldsymbol{a}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$;
      end
    end
  end
end
return $\gamma_{\boldsymbol{a}}\left(n^{(X)}[0, L^{(X)}], n^{(Y)}[0, L^{(Y)}], n^{(Z)}[0, L^{(Z)}]\right)$;

Algorithm 6: The constrained CYK algorithm for three sequences X, Y, Z. Ensemble states $\boldsymbol{a}$ in the iteration over states are sorted in Inside fill order with `Emit` states first, then `Null` states in reverse topological order.

Input: state $\boldsymbol{a}, n^{(X)}, n^{(Y)}, n^{(Z)}$, intermediate CYK matrix γ
emitProb $\leftarrow 0$;
foreach $\boldsymbol{b} : \exists\, \boldsymbol{a} \rightarrow \boldsymbol{b}$ **do**
  emitProb $= \max\left(\text{emitProb}, P\left(\boldsymbol{a} \rightarrow \boldsymbol{b}\right) \gamma_{\boldsymbol{b}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)\right)$;
end
foreach $\boldsymbol{b} : \exists\, \boldsymbol{a} \rightarrow \boldsymbol{l}\,\boldsymbol{b}\,\boldsymbol{r}$ **do**
  if $c_{in}\left(\boldsymbol{b}; n^{(X)}\right) \notin \mathscr{F}^{(X)}$ *or* $c_{in}\left(\boldsymbol{b}; n^{(Y)}\right) \notin \mathscr{F}^{(Y)}$ *or* $c_{in}\left(\boldsymbol{b}; n^{(Z)}\right) \notin \mathscr{F}^{(Z)}$ **then** next;
  emitProb $= \max\left(\text{emitProb}, P\left(\boldsymbol{a} \rightarrow \boldsymbol{l}\,\boldsymbol{b}\,\boldsymbol{r}\right) \gamma_{\boldsymbol{b}}\left(c_{in}\left(\boldsymbol{b}; n^{(X)}\right), c_{in}\left(\boldsymbol{b}; n^{(Y)}\right), c_{in}\left(\boldsymbol{b}; n^{(Z)}\right)\right)\right)$;
end
return emitProb;

Algorithm 7: Subroutine calcTransEmitProb() for the CYK algorithm. $\boldsymbol{a}$ and $\boldsymbol{b}$ are ensemble states; $\boldsymbol{l}$ and $\boldsymbol{r}$ are left and right terminal emissions.

Input: sequences X, Y, Z, Inside matrix α
foreach $n^{(X)} \in \mathscr{F}^{(X)}$ **do** `/* outside-inside sorted */`
  foreach $n^{(Y)} \in \mathscr{F}^{(Y)}$ **do** `/* outside-inside sorted */`
    foreach $n^{(Z)} \in \mathscr{F}^{(Z)}$ **do** `/* outside-inside sorted */`
      foreach *state* $\boldsymbol{b}$ **do**
        bifurcProb $\leftarrow 0$;
        foreach $\left(n_O^{(X)}, n_L^{(X)}\right) \in b_{out,L}\left(n^{(X)}\right)$ **do**
          foreach $\left(n_O^{(Y)}, n_L^{(Y)}\right) \in b_{out,L}\left(n^{(Y)}\right)$ **do**
            foreach $\left(n_O^{(Z)}, n_L^{(Z)}\right) \in b_{out,L}\left(n^{(Z)}\right)$ **do**
              bifurcProb += `calcLBifurcProb` $(\boldsymbol{b}; \cdot)$;
            end
          end
        end
        foreach $\left(n_O^{(X)}, n_R^{(X)}\right) \in b_{out,R}\left(n^{(X)}\right)$ **do**
          foreach $\left(n_O^{(Y)}, n_R^{(Y)}\right) \in b_{out,R}\left(n^{(Y)}\right)$ **do**
            foreach $\left(n_O^{(Z)}, n_R^{(Z)}\right) \in b_{out,R}\left(n^{(Z)}\right)$ **do**
              bifurcProb += `calcRBifurcProb` $(\boldsymbol{b}; \cdot)$;
            end
          end
        end
        $\beta_{\boldsymbol{b}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right) \leftarrow$ `calcTransEmitProb` $\left(\boldsymbol{b}; n^{(X)}, n^{(Y)}, n^{(Z)}\right)$;
        $\beta_{\boldsymbol{b}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$ += bifurcProb ;
        store $\beta_{\boldsymbol{b}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$;
      end
    end
  end
end

Algorithm 8: The constrained Outside algorithm for three sequences X, Y, Z. Ensemble states $\boldsymbol{a}$ in the iteration over states are sorted in Outside fill order with `Emit` states first, then `Null` states in topological order.

Input: state $\boldsymbol{b}, n^{(X)}, n^{(Y)}, n^{(Z)}$, intermediate Outside matrix β
emitProb $\leftarrow 0$;
foreach $\boldsymbol{a} : \exists\, \boldsymbol{a} \rightarrow \boldsymbol{b}$ **do**
  emitProb += $P(\boldsymbol{a} \rightarrow \boldsymbol{b})\ \beta_{\boldsymbol{a}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$;
end
foreach $\boldsymbol{a} : \exists\, \boldsymbol{a} \rightarrow \boldsymbol{l}\,\boldsymbol{b}\,\boldsymbol{r}$ **do**
  if $c_{out}\left(\boldsymbol{b}; n^{(X)}\right) \notin \mathscr{F}^{(X)}$ *or* $c_{out}\left(\boldsymbol{b}; n^{(Y)}\right) \notin \mathscr{F}^{(Y)}$ *or* $c_{out}\left(\boldsymbol{b}; n^{(Z)}\right) \notin \mathscr{F}^{(Z)}$ **then** next;
  emitProb += $P(\boldsymbol{a} \rightarrow \boldsymbol{l}\,\boldsymbol{b}\,\boldsymbol{r})\ \beta_{\boldsymbol{a}}\left(c_{out}\left(\boldsymbol{b}; n^{(X)}\right), c_{out}\left(\boldsymbol{b}; n^{(Y)}\right), c_{out}\left(\boldsymbol{b}; n^{(Z)}\right)\right)$;
end
return emitProb;

Algorithm 9: Subroutine calcTransEmitProb() for the Outside algorithm. $\boldsymbol{a}$ and $\boldsymbol{b}$ are ensemble states; $\boldsymbol{l}$ and $\boldsymbol{r}$ are left and right terminal emissions.

Input: sequences X, Y, Z
pushdown stack coordinateStack; `/* hold (state, subsequence triplet) pairs */`
$\boldsymbol{a} \leftarrow$ `Start`; `/* current ensemble state */`
$n^{(X)} \leftarrow N^{(X)}$; `/* current` X `subsequence;` $N^{(X)}$ `is the outermost subsequence */`
$n^{(Y)} \leftarrow N^{(Y)}$; `/* current` Y `subsequence;` $N^{(Y)}$ `is the outermost subsequence */`
$n^{(Z)} \leftarrow N^{(Z)}$; `/* current` Z `subsequence;` $N^{(Z)}$ `is the outermost subsequence */`
clear coordinateStack;
begin main loop:
  output current state $\boldsymbol{a}$ and subsequence triplet $\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right)$;
  if $\boldsymbol{a}$ *is the* `End` *state* **then** `/* end of a parse subtree */`
    if coordinateStack *is empty* **then return** ; `/* end of the parse tree */`
    pop $\left(\boldsymbol{a}, n^{(X)}, n^{(Y)}, n^{(Z)}\right)$ from coordinateStack ;
    goto main loop ;
  else if $\boldsymbol{a}$ *is a bifurcation state* **then** `/* bifurcation` $\boldsymbol{a} \rightarrow \boldsymbol{cb}$ `*/`
    select $\left(n_L^{(X)}, n_R^{(X)}\right) \in b_{in}\left(n^{(X)}\right)$, $\left(n_L^{(Y)}, n_R^{(Y)}\right) \in b_{in}\left(n^{(Y)}\right)$, $\left(n_L^{(Z)}, n_R^{(Z)}\right) \in b_{in}\left(n^{(Z)}\right)$
    such that
    $\gamma_{\boldsymbol{a}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right) = \gamma_{\boldsymbol{a}}\left(n_L^{(X)}, n_L^{(Y)}, n_L^{(Z)}\right)\gamma_{\boldsymbol{a}}\left(n_R^{(X)}, n_R^{(Y)}, n_R^{(Z)}\right)$;
    push $\left(\boldsymbol{c}, n_R^{(X)}, n_R^{(Y)}, n_R^{(Z)}\right)$ onto coordinateStack ;
    $\boldsymbol{a} \leftarrow \boldsymbol{b}$;
    $n^{(X)} \leftarrow n_L^{(X)}$;
    $n^{(Y)} \leftarrow n_L^{(Y)}$;
    $n^{(Z)} \leftarrow n_L^{(Z)}$;
    goto main loop ;
  else `/* Emit or Null state */`
    $m^{(X)} \leftarrow c_{in}\left(\boldsymbol{b}; n^{(X)}\right)$;
    $m^{(Y)} \leftarrow c_{in}\left(\boldsymbol{b}; n^{(Y)}\right)$;
    $m^{(Z)} \leftarrow c_{in}\left(\boldsymbol{b}; n^{(Z)}\right)$;
    select $\boldsymbol{b} \in \{b : \exists\, \boldsymbol{a} \rightarrow \boldsymbol{lbr}\}$
    such that
    $\gamma_{\boldsymbol{a}}\left(n^{(X)}, n^{(Y)}, n^{(Z)}\right) = P(\boldsymbol{a} \rightarrow \boldsymbol{lbr})\gamma_{\boldsymbol{b}}\left(m^{(X)}, m^{(Y)}, m^{(Z)}\right)$;
    $\boldsymbol{a} \leftarrow \boldsymbol{b}$;
    $n^{(X)} \leftarrow m^{(X)}$;
    $n^{(Y)} \leftarrow m^{(Y)}$;
    $n^{(Z)} \leftarrow m^{(Z)}$;
    goto main loop ;
end

Algorithm 10: The constrained CYK traceback algorithm for three sequences X, Y, Z.