JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 13, Number 5, 2006

Pp. 1077–1100

Grammatical Representations of Macromolecular Structure

DAVID CHIANG,[1] ARAVIND K. JOSHI,[2] and DAVID B. SEARLS[3]

ABSTRACT

Since the first application of context-free grammars to RNA secondary structures in 1988, many researchers have used both *ad hoc* and formal methods from computational linguistics to model RNA and protein structure. We show how nearly all of these methods are based on the same core principles and can be converted into equivalent approaches in the framework of tree-adjoining grammars and related formalisms. We also propose some new approaches that extend these core principles in novel ways.

Key words: computational linguistics, formal grammars, tree-adjoining grammars.

1. INTRODUCTION

COMPUTATIONAL LINGUISTIC METHODS, broadly construed, have been applied to molecular biology in two general ways, which we classify as *textual* and *structural* (Searls, 1999). Textual approaches bear mainly on the actual string content of biological sequences, whereas structural approaches deal mainly with the interactions between sequence elements in folded structures.

Examples of the textual approach include the use of regular expressions to specify recurring motifs, or of grammars that capture gene structures as assemblages of codons, signal sequences, and other lexical elements. The latter application demonstrates how linguistic methods (whether based on grammars or their cognate automata) allow such primary sequence elements to be collected in rule-based fashion into flexible hierarchical descriptions that both enforce global constraints such as reading frame and provide a useful compositional framework for heuristic and statistical discrimination of, for instance, coding versus non-coding segments (Dong and Searls, 1994). Though textual methods may assign hierarchical structures to sequences as a convenient and meaningful abstraction, these structures are not meant to correspond in any way with physical, three-dimensional structure. Textual approaches are most closely aligned with conventional sequence analysis, and in fact many tasks such as pairwise alignment can be recast in automata-theoretic terms (Searls, 1999). Among the most popular textual tools have been hidden Markov models (HMMs), whose use was pioneered in speech processing applications, but which have been applied with great success in such biological arenas as classification and recognition of protein motifs (Durbin *et al.*, 1999) and in gene-finding algorithms that entail both complex domain models and advanced statistical methods (Burge and Karlin, 1997; Reese *et al.*, 2000).

[1]Information Sciences Institute, University of Southern California, Marina del Rey, California.
[2]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pennsylvania.
[3]Bioinformatics Division, Genetics Research, GlaxoSmithKline Pharmaceuticals, King of Prussia, Pennsylvania.

The structural approach deals more with interactions such as base-pairing in RNA secondary structure and side-chain interactions in native proteins, particularly hydrogen bonding in α-helices and β-sheets. These relationships between string positions potentially separated by long distances are loosely analogous to syntactic *dependencies*: for example, just as in RNA two base-paired nucleotides must be complementary, so in English, a verb and its dependent subject must agree. The emphasis is thus on the inherent ability of grammars or other tools adequately to model the variety of patterns of interaction observed in nature. Linguists typically seek formalisms that are just sufficiently elaborate and powerful to encompass the range of phenomena under study, but not more so (Berwick, 1984); representational parsimony helps to ensure that the related algorithmic and mathematical challenges are most tractable, but more importantly, that clarifying generalizations are most likely to emerge from the models. One hopes that the basic formal model will capture and dictate as much of the domain as possible, neither overgenerating nor undergenerating the phenomena of interest, before having to resort to overlying ad hoc restrictions or extensions. Ultimately, such insights may also support a convergence of structural models and textual tools, as in the extension of HMMs to more powerful stochastic context-free grammars (SCFGs) to recognize RNA secondary structure motifs (Sakakibara *et al.*, 1994).

A number of recent attempts have been made to model macromolecular structures in this manner that have been founded more or less upon the foundations of formal language theory and computational linguistics. In this paper, these systematic approaches will be reviewed, consolidated with mainstream computational linguistic formalisms, and in several cases significantly extended. After some theoretical background, we review several methods for modeling RNA pseudoknots (Uemura *et al.*, 1999; Rivas and Eddy, 2000; Cai *et al.*, 2003; Brown and Wilson, 1996), unifying all but one in the framework of *linear context-free rewriting systems* (Vijay-Shanker *et al.*, 1987; Weir, 1988), and add a proposal for modeling other limited kinds of RNA tertiary interactions. We then turn to protein structure, reviewing a method for modeling protein β-sheets (Abe and Mamitsuka, 1997) and propose a quite different method that promises to be considerably more efficient. Finally, we discuss a method for modeling α-helix bundles (Waldispühl and Steyaert, 2005) and propose some approaches of our own.

2. PRELIMINARIES

We begin by elaborating somewhat on linguistic notions of structure, strong generative capacity, and locality as they relate to macromolecules. The reader is referred to the extensive linguistics literature for a more detailed development.

2.1. Structure

Formal language theory views languages as sets of finite strings over finite alphabets, and it is natural to represent biological sequences as strings over a set Σ of monomers (nucleotides or amino acids), that is, as members of

$$\Sigma^* = \{a_1 \cdot a_2 \cdot a_3 \cdots a_n \mid a_i \in \Sigma\}$$

where the concatenation operator $\cdot$ (usually omitted) can be said to model the covalent bonds of the polymer.

It might have been thought that linguistics, with its linear string orientation, had little to offer to structural biology, which represents macromolecules in terms of their three-dimensional coordinates, focusing on their actual spatial arrangement or even space-filling properties. Yet, not only does linguistics concern itself with structure in various senses, but structural biology also employs formalisms at higher levels of abstraction, such as the classification of secondary and supersecondary structural motifs, the use of lattice models of folding, and the like. In particular, reduced representations such as polymer graphs or contact maps, reminiscent of linguistic dependency structures, may be viewed as an abstracted representation of structure. For these purposes, we thus conceive of a macromolecular structural description as a pair consisting of a finite string and a set of pairs of positions in that string.

$$\langle p, s \rangle \text{ where } p \in \Sigma^*, s \subseteq \{\langle i, j \rangle \mid 1 \leq i, j \leq |p|, i \neq j\} \tag{1}$$

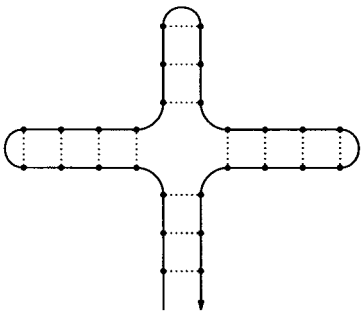

FIG. 1. Example RNA secondary structure.

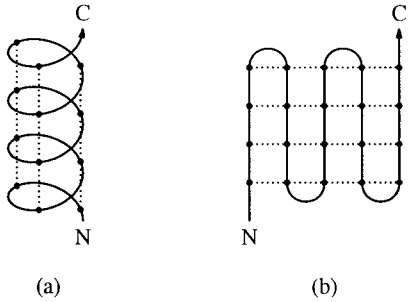

FIG. 2. Example protein secondary structures: (**a**) α-helix. (**b**) β-sheet.

In the case of nucleic acids, the pair may be seen to comprise a primary (p) and a secondary (s) structure; the latter term is used somewhat differently in proteins, but in either case the pair can also be viewed as the polymer (p) and its structure (s). In linguistic terms, p is a strictly lexical description, whereas s may reflect (though perhaps not fully describe) syntactic relationships manifesting as dependencies. In chemical terms, p models only the covalent bonds of the linear polymer backbone, while s models the non-covalent "cross-linking" interactions contributing to structure,[1] which may include base-pairing, other forms of hydrogen bonding, hydrophobic or charge interactions, shape complementarity, and so forth—or simply self-contact, for that matter, though the linguistic notion implies mutual influence of some sort.

As suggested previously, textual approaches generally deal with p exclusively, while structural approaches in the abstract need only consider s. In either case, we seek grammars (or other formal systems and procedures) that can be shown to generate a corpus of instances observed in nature, and to do so in such a way as to offer useful generalizations, explanations, classifications, and frameworks supporting further analysis.

It is easy to see ways to extend (1) to encompass typing of dependencies or additional annotation such as directionality, cheirality, and the like, as has been done in quasi-linguistic systems along the lines of TOPS diagrams of abstract protein structures (Westhead *et al.*, 1999). This is very much in the linguistic tradition, where dependencies may be richly typed and annotated.

However, our formulation is simpler than linguistic dependency structure in some important ways. Syntactic dependencies are usually asymmetric (directed), involving a *head* and a *dependent*; our molecular dependencies are symmetric (undirected). Syntactic dependency graphs are often tree-structured, though not always (Mel'čuk, 1988); our molecular dependencies need not be.

[1]A possible exception would be disulfide bonds between cysteine residues of certain proteins, which are covalent, though distinct from the peptide bonds of the polymer backbone by virtue of being post-translational and labile depending on redox conditions.

2.2. Strong generative capacity

Linguists consider a sentence structure to be a syntactic organization (whether of constituency, dependency, or both) underlying a lexical string. Such structure can be generated by grammars, which not only specify sets of strings but produce derivation trees reflecting the order and pattern of application of rewrite rules. The *weak generative capacity* (WGC) of a grammar is the set of strings it generates—the p components in (1). In natural language, one hopes for a formalism to generate precisely the sentences in the language that are acceptable to a native speaker of that language, though no such test is readily available for sequence languages other than sampling and it would be difficult to judge whether they are undergenerated or overgenerated (another reason to focus on s rather than p).

A more meaningful concept in either case is the *strong generative capacity* (SGC) of a grammar or grammar formalism, which describes the actual structures that can be generated, for instance, different groupings of words into phrase structures. This is a better measure because more than one structure is possible for a given string, and even if a string is accepted by a grammar, it may be the case that the grammar assigns it an inappropriate structure. Indeed, there are cases where more than one possible structure is meaningful for a given sequence, as seen, for instance, with alternative secondary structure in nucleic acids, and we want our grammars to capture such phenomena.

Though the term "strong generative capacity" is sometimes applied narrowly to phrase structures, it applies more generally to structures of any kind; for our purposes, we are concerned with the structures defined in (1). In this paper, then, we use the term "strong generative capacity" to refer more precisely to the set of structures of the form (1) that can be generated by a grammar (in the manner defined below). This notion has been previously introduced as *derivational generative capacity* (Becker *et al.*, 1991).

2.3. Locality

Beginning with the first demonstration that generative grammars could model nucleic acid stem-and-loop secondary structure (Searls, 1988), a particularly attractive feature of this approach has been the observation that the derivation trees actually resemble the usual graphical depiction of those structures. For example, a parse of a tRNA sequence derived from a simple context-free grammar (CFG) can overlay the actual cloverleaf structure as illustrated in textbooks. It should be borne in mind that such drawings are themselves two-dimensional projections and simplifications of the literal structures, but the fact remains that the formalism is bringing elements into apposition in derivational structures that are also in contact in the corresponding literal structures.

This property is related to the establishment of dependencies by grammars. Consider the following context-free grammar (CFG) for RNA sequences:

$$\begin{aligned} \mathrm{S} &\rightarrow \mathrm{Z} \\ \mathrm{X} &\rightarrow \mathrm{aZu} \mid \mathrm{uZa} \mid \mathrm{cZg} \mid \mathrm{gZc} \\ \mathrm{Y} &\rightarrow \mathrm{aY} \mid \mathrm{uY} \mid \mathrm{cY} \mid \mathrm{gY} \mid \epsilon \\ \mathrm{Z} &\rightarrow \mathrm{YXZ} \mid \mathrm{Y} \end{aligned} \tag{2}$$

This grammar generates stem structures with the rule for X, and unpaired bases with the rule for Y, interspersing them arbitrarily via Z. A number of variations on this secondary structure CFG have been introduced, with or without unpaired bases allowed, which are weakly and in many cases strongly equivalent to this one (Searls, 1999); another related approach, which however is algebraic rather than grammar-based, deals with properties of strings of properly-nested parentheses called Motzkin words, and has been applied to the combinatorics of RNA secondary structure (Nebel, 2002; Viennot and de Chaumont, 1983). As noted, a derivation of a string w, represented as a tree (Fig. 3), has the same shape as a secondary structure of w (Fig. 1); this is because the grammar is written so that only complementary bases appear in the same rule for X, and CFG derivation trees have the convenient property that symbols from the same rule appear next to each other in the tree. Thus, formal locality (within rules) corresponds to spatial locality (in derivation trees, thence in secondary structure).

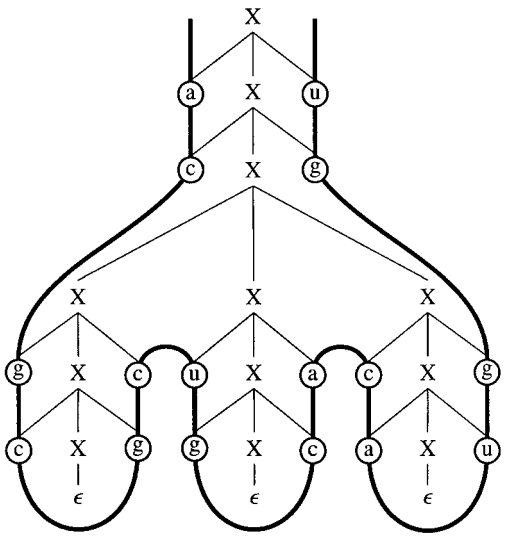

FIG. 3. Example CFG derivation of RNA secondary structure, with superimposed primary structure. Nonterminal symbols other than X are suppressed for clarity.

We have also explicitly marked the base-pairing dependency within the grammar, as a dotted line. This notation serves two purposes. First, even if we do not represent CFG derivations as trees, we can still use them to describe secondary structures. For example, we can represent CFG derivations as rewritings of sentential forms, preserving the explicitly marked base pairings:

$$\begin{aligned} \mathrm{X} &\overset{*}{\Rightarrow} \mathrm{aXu} \\ &\overset{*}{\Rightarrow} \mathrm{acXgu} \\ &\overset{*}{\Rightarrow} \mathrm{acXXXgu} \\ &\overset{*}{\Rightarrow} \mathrm{acgXcXXgu} \end{aligned} \tag{3}$$

The base pairings get "stretched" in the sentential forms, and capture the secondary structure even though the sentential forms are not graphically folded up like real molecules. Thus, the fact that CFG derivation trees place items from the same local domain next to each other is a convenient property but not an essential one.

This distinction becomes more important as we move to higher formalisms. For example, it will be seen that in a type of grammar called *tree-adjoining grammar* (to be defined below), parts of rules can be stretched arbitrarily far apart during the derivation process.[2] But even if formal locality cannot correspond to spatial locality in our drawings of derivations, they can still correspond to spatial locality in real molecules. In other words, derivations can still describe molecules even if their drawings don't look very much like them, in effect sidestepping the shortcomings of the two-dimensional projection represented by both the drawings and the derivation trees.

[2]However, Rogers (2003) explores the use of three-dimensional trees to represent derivations of tree-adjoining grammars, and higher-dimensional trees for still more complex formalisms. In a tree-adjoining grammar defined on three-dimensional trees, there is no stretching.

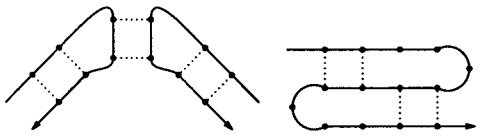

FIG. 4. Example RNA tertiary structures: (**a**) Kissing hairpins. (**b**) Pseudoknot.

The second purpose of explicitly marking self-contacts in the grammar is that it provides more flexibility. In our original treatment (Searls, 1992) and that of Uemura *et al.* (1999), any pair of monomers generated by the same rule were modeled as being in contact. But if self-contacts are explicitly marked in the grammar, then the grammar writer is free to decide whether pairs of monomers in a rule should be modeled as being in contact. This leaves us with the following locality constraint: two nonadjacent monomers *may* be modeled as being in contact only if their corresponding symbols were generated in the same derivation step. All uses of formal grammars to model biological molecules of which we are aware (and of which many are elaborated below) are based on this principle, though with variations and sometimes only implicitly. In addition to the examples mentioned above, Rivas and Eddy (2000) use diagrams reminiscent of Joshi's links (Joshi, 1985). The model of Abe and Mamitsuka (1997) does not allow for self-contacts to be specified on elementary structures, with the perhaps undesirable result that a single derivation can correspond to multiple structures. Chen and Dill (1998) do not use a grammar at all, but their model can be recast as a CFG (Chiang and Joshi, 2002; cf. Mauri *et al.*, 1999); they specify self-contacts in a manner also similar to Joshi's links.

Does CFG have enough SGC for modeling biological molecules? The SGC of CFG is commonly characterized by saying that CFG cannot generate crossing dependencies, which is not strictly true, since even a single production can have crossing dependencies:

$$\mathrm{X} \rightarrow \overset{\frown}{\mathrm{a}\,\mathrm{b}\,\mathrm{c}\,\mathrm{d}} \tag{4}$$

Nevertheless, there are patterns of crossing self-contacts occurring in nature that are provably not generable by CFG (including Chen and Dill's model). For example, helices involve unbounded series of short crossing self-contacts. RNA tertiary structures involve long-distance crossing self-contacts, for example, *kissing hairpins* and *pseudoknots* (Fig. 4). Finally, protein β-sheets and α-helix bundles involve patterns of crossing self-contacts that are well beyond the power of CFG. In what follows we will examine in some detail formalisms that are beyond context-free, and which address specific non-context-free phenomena in biology.

3. RNA TERTIARY STRUCTURE

Pseudoknots (Fig. 4b) are the archetypal feature of so-called non-orthodox nucleic acid secondary structure (that is, structure with crossing dependencies), and have been taken as a challenge by both algorithm developers and those seeking formal representations of many sorts. We now explore several of these formalisms, taking the opportunity to describe them in some mathematical detail.

3.1. Tree-adjoining grammars

For pseudoknots, Uemura *et al.* (1999) propose moving beyond CFGs to *tree-adjoining grammars* or TAGs (Joshi *et al.*, 1975).[3] Figure 6 shows an example TAG, which is a simplified version of Uemura

[3]We deal here with TAGs as they relate to biological sequences; for a more general overview of TAG with applications to natural language, see the computational linguistics literature (Joshi and Schabes, 1997; Joshi, 2004).

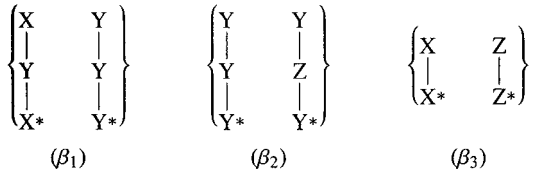

FIG. 5. Example multicomponent TAG elementary tree sets.

et al.'s grammar. Just as a CFG has a set of productions, a TAG has a set of *elementary trees*, here $\{\alpha, \beta_1, \beta_2\}$. The tree α is known as an *initial tree*, and the trees β_1 and β_2 are known as *auxiliary trees*, being distinguished by a special frontier node called the *foot node*, marked with a $*$. The root node and foot node must have the same label. The path from the root node to the foot node of an auxiliary tree is called its *spine*.

Just as the basic operation of CFG is symbol rewriting, the basic operation of TAG is node rewriting, or *adjunction*. Any node η that is not marked NA (for *no adjunction*) and is not a foot node can be rewritten by adjunction. In this operation η is removed, splitting the tree into an upper half and a lower half, and an auxiliary tree β takes its place, its root attaching to the upper half where η used to be, and its foot attaching to the lower half. Figure 7 illustrates a derivation starting with α, adjoining in β_1, adjoining in β_1 again, and then adjoining β_2.

Note that the links generated by this derivation cross, and a link may cross a potentially unbounded number of other links. Intuitively, the added power of TAG comes from the fact that CFG is limited to specifying interactions between sister nodes, whereas TAG can specify interactions between nodes on different levels, indeed, on levels that are arbitrarily far apart (because of adjunction).

We will make use below of an extension of TAG called *set-local multicomponent TAG* (Weir, 1988). Multicomponent TAGs are TAGs whose elementary structures are sets of elementary trees. The basic operation is the simultaneous adjunction of all the trees in a set. In *set-local* multicomponent TAG, all the trees must compose into the same elementary tree set. For example, Fig. 5 shows some multicomponent TAG elementary tree sets. In a set-local multicomponent TAG, β_2 would be able to adjoin into β_1, by adjoining the first component into the first component and the second component into the second component. But β_3 would not be able to adjoin into the result, because the X and Z nodes come from different elementary tree sets.

3.2. Linear TAG

Uemura's grammar belongs to a subclass of TAG which they call *simple linear TAG*, in which every auxiliary tree has at most one node at which adjunction is allowed, and this node lies on its spine. This subclass is parseable in $\mathcal{O}(n^4)$ time. They also define a larger subclass, called *extended simple linear TAG*, in which a second adjunction is allowed off the spine. This allows them to write a more versatile grammar which we do not reproduce here. This subclass of TAG is parseable in $\mathcal{O}(n^5)$ time and has been conjectured (Kato *et al.*, 2004), we believe correctly, to be equivalent to the TAG restriction of Satta and Schuler (1998).

One of the more complicated RNA structures known to occur in nature is the hepatitis delta virus ribozyme (Hilbers *et al.*, 1998), an abstract representation of which is shown in Fig. 8a. This structure can

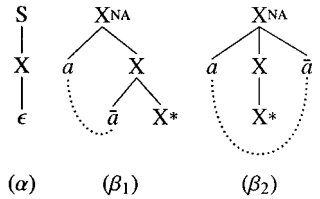

FIG. 6. TAG fragment for pseudoknots, adapted from the grammar of Uemura *et al.* (1999). Here and elsewhere, $\bar{a}$ stands for the complementary base of a.

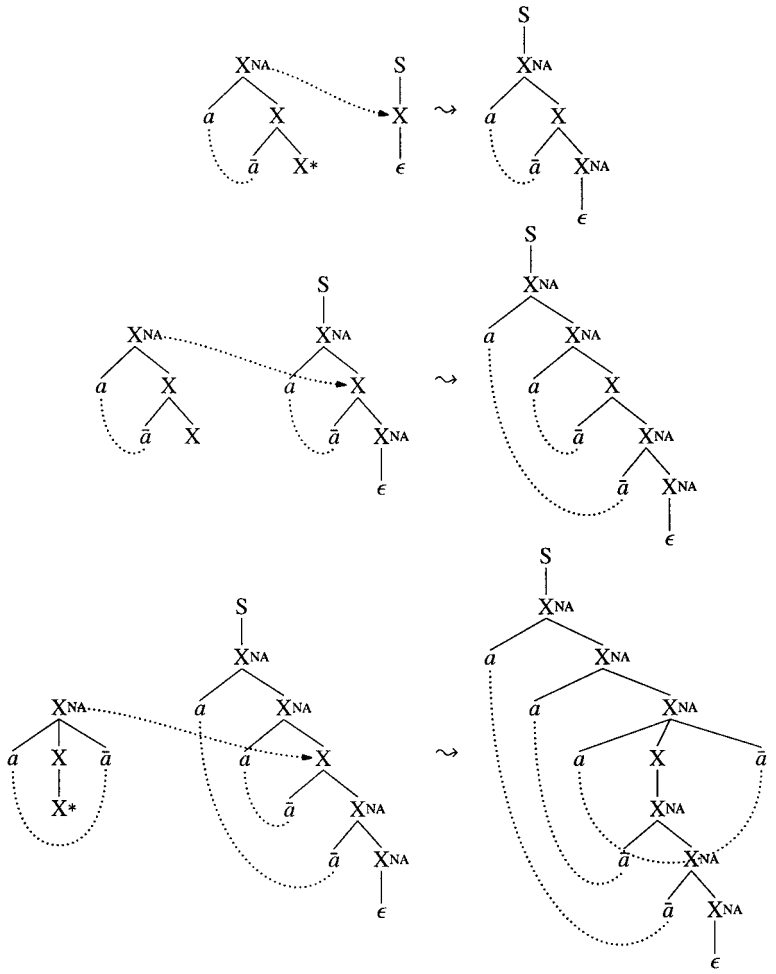

FIG. 7. Example derivation of grammar of Fig. 6.

be generated by the simple linear TAG of Fig. 8c with the derivation of Fig. 8b, and therefore it can be generated by the following two approaches as well.

3.3. Crossed-interaction grammars

Rivas and Eddy (2000) define a formalism called crossed-interaction grammar (CIG) to describe RNA pseudoknots. This formalism appears to be equivalent to *linear context-free rewriting systems* or LCFRS (Vijay-Shanker *et al.*, 1987; Weir, 1988), a class of grammars that has been independently rediscovered quite a number of times (Rambow and Satta, 1999); the set-local multicomponent TAGs defined above are another equivalent example.

A CIG has two modules, a set of context-free productions and a set of *rearrangement rules*. The productions work as in CFG, except that there is a *hole string* $\wedge$ and a set of *special nonterminals*. As far as the productions are concerned, these are all ordinary terminal symbols.

The rearrangement rules apply after the productions. Unlike the context-free productions, these are general rewrite rules, the only restriction being that no special nonterminal may appear on the right-hand side. Furthermore, a grammar may have an infinite number of such rules.

However, their rearrangement rules follow certain implicit conventions, which greatly reduce the power of the system. We make the following six observations:

1. The infinite set of rearrangement rules is defined by a finite set of rule schemata, with variables that range freely over terminal strings.

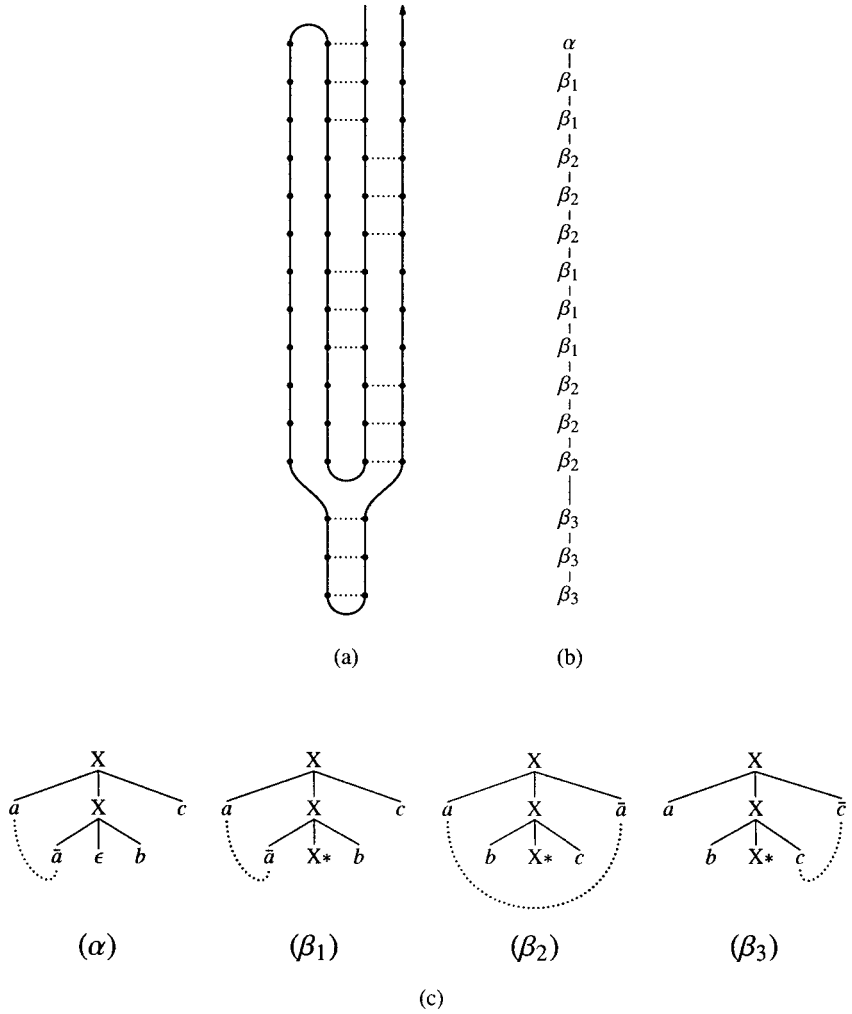

FIG. 8. Pseudoknot in hepatitis delta virus ribozyme. (**a**) Abstract representation of structure. (**b**) Derivation by simple linear TAG. (**c**) Elementary trees used in derivation.

The schemata, rather than their instantiations, should be thought of as the rules. Below we will use "rearrangement rule" and "rearrangement rule schema" interchangeably.

2. No two rearrangement rules have the same left-hand side.
3. Every variable on the left-hand side is delimited by a special nonterminal or the hole string.

Therefore, no two left-hand sides can be instantiated to the same string.

4. Every variable on the right-hand side appears on the left-hand side.

Therefore, every instantiation of a left-hand side uniquely determines an instantiation of the right-hand side.

5. Parentheses are ***special*** special nonterminals: every left-hand side is surrounded by parentheses, and parentheses appear nowhere else.

Thus, the rearrangement phase is deterministic, because only an innermost parenthesized string can be rewritten. Therefore, we can view the special nonterminals as infix string operators, and the rearrangement

rules as their definitions. This second phase, then, is nothing more than the evaluation of the output of the first phase. Furthermore,

6. Every variable on the left-hand side appears exactly once on the right-hand side.

Thus, the rearrangement rules, as their name suggests, do not copy or delete their arguments, but only perform rearrangements of a finite number of chunks of the string.

In light of these observations, we now give a more precise definition of CIG.

Definition 1. *Given two finite alphabets Σ and F, define $F(\Sigma)$ to be the smallest set which satisfies:*

- $\Sigma \subseteq F(\Sigma)$.
- *If* $w_1, \ldots, w_n \in F(\Sigma), n \geq 0$ *and* $f \in F$, *then* $f(w_1, \ldots, w_n) \in F(\Sigma)$.

For example, if $\Sigma = \{A, B\}$ *and* $F = \{f, g\}$, *then* $f(A, g(B, f)) \in F(\Sigma)$.

Definition 2. *A* crossed-interaction grammar *is a tuple* $\langle V, F, P, S, X, T, R \rangle$, *where:*

- V *is a finite set of* nonterminal symbols*;*
- F *is a finite set of* function symbols, *which correspond to the special nonterminals except that we use a simpler prefix notation;*
- P *is a finite set of* productions, *each of the form* $A \rightarrow w$, *where* $A \in V$ *and* $w \in F(V)$*;*
- $S \in V$ *is the* start symbol*;*
- T *is a finite set of* terminal symbols *and* X *is a finite set of* variables,
- R *is a finite set of* rearrangement rules, *each of the form*

$$f(\langle x_{11}, \ldots, x_{1m_1} \rangle, \ldots, \langle x_{n1}, \ldots, x_{nm_n} \rangle) = \langle y_1, \ldots, y_m \rangle,$$

where

- $f \in F$, $x_{ij} \in X$, *and* $y_i \in (X \cup T)^*$, *and*
- *every variable that appears in the rule appears exactly once on the left-hand side and exactly once on the right-hand side.*

Thus the rearrangement rules define functions on tuples of strings $\langle w_1, \ldots, w_n \rangle$, *which correspond to gapped strings* $w_1 \wedge \cdots \wedge w_n$.

The derivation process is straightforward: the productions operate as in a CFG, beginning with S and rewriting until a string without nonterminals (i.e., a member of $F(\emptyset)$) is produced. Then the resulting expression is evaluated according to the rearrangement rules.

As an example, consider the following grammar:

$$\begin{aligned}
S &\rightarrow m(T) \\
T &\rightarrow \times(T, \wedge(a(), b())) \mid \wedge(e(), e()) \\
m(\langle x_1, x_2 \rangle) &= \langle x_1 x_2 \rangle \\
\times(\langle x_1, x_2 \rangle, \langle y_1, y_2 \rangle) &= \langle x_1 y_1, x_2 y_2 \rangle \\
\wedge(\langle x \rangle, \langle y \rangle) &= \langle x, y \rangle \\
a() &= \langle \mathrm{a} \rangle \\
b() &= \langle \mathrm{b} \rangle \\
e() &= \langle \epsilon \rangle
\end{aligned}$$

This grammar generates the language $a^n b^n$ in such a way that the ith a and ith b are generated in the same derivation step:

$$\begin{aligned}
S &\Rightarrow m(T) \\
&\Rightarrow m(\times(T, \wedge(a(), b()))) \\
&\Rightarrow m(\times(\times(T, \wedge(a(), b())), \wedge(a(), b()))) \\
&\Rightarrow m(\times(\times(\wedge(e(), e()), \wedge(a(), b())), \wedge(a(), b()))) \\
&= m(\times(\times(\wedge(\langle\epsilon\rangle, \langle\epsilon\rangle), \wedge(\langle\mathrm{a}\rangle, \langle\mathrm{b}\rangle)), \wedge(\langle\mathrm{a}\rangle, \langle\mathrm{b}\rangle))) \\
&= m(\times(\times(\langle\epsilon, \epsilon\rangle, \langle\mathrm{a}, \mathrm{b}\rangle), \langle\mathrm{a}, \mathrm{b}\rangle)) \\
&= m(\times(\langle\mathrm{a}, \mathrm{b}\rangle, \langle\mathrm{a}, \mathrm{b}\rangle)) \\
&= m(\langle\mathrm{aa}, \mathrm{bb}\rangle) \\
&= \langle\mathrm{aabb}\rangle
\end{aligned}$$

An LCFRS has the same form as above except that all productions have the form $A \rightarrow f(A_1, \ldots, A_n)$. It is not hard to show that the two definitions are equivalent.

Following Rambow and Satta (1999), define the *rank* of a function to be its number of arguments, and its *fan-out* to be the size of the tuples it operates on. We then say that a CIG has rank r if none of its rearrangement rules define a function of rank greater than r, and fan-out f if none of its rearrangement rules define a function with fan-out greater than f. Rivas and Eddy limit both the rank and fan-out of CIG to two. They use the following rearrangement rules:

$$\begin{aligned}
\times(\langle x_1, x_2\rangle, \langle y_1, y_2\rangle) &= \langle x_1 y_1, x_2 y_2\rangle \\
\times_L(\langle x_1, x_2\rangle, \langle y_1, y_2\rangle) &= \langle y_1 x_1 y_2, x_2\rangle \\
\times_R(\langle x_1, x_2\rangle, \langle y_1, y_2\rangle) &= \langle x_1, y_1 x_2 y_2\rangle \\
\supset(\langle x_1, x_2\rangle, \langle y_1, y_2\rangle) &= \langle x_1 y_1, y_2 x_2\rangle
\end{aligned}$$

It can be shown that a CIG/LCFRS with rank r and fan-out f can be parsed in $\mathcal{O}(n^{(r+1)f})$ time. Thus, this grammar can be parsed in $\mathcal{O}(n^6)$ time, like TAG. However, the class of grammars that satisfies this restriction is more powerful than TAG, because the operations $\times$, $\times_L$, $\times_R$ are not permitted in general by TAG.

For example, the operators $\times_L$ and $\times_R$ allow structures like the one shown in Fig. 9, which cannot be generated by TAG (Akutsu, 2000). However, such structures are not known to occur in nature. Another example comes from Weir (1988), who already identified the rank two, fan-out two class of languages and gave the language $a^n b^n a^m b^m c^n d^n c^m d^m$ as an example of a language in this class but beyond TAG. Below is a CIG with rank two and fan-out two that generates this language:

$$\begin{aligned}
W &\rightarrow m(\times(W_H, W_H)) \\
W_H &\rightarrow \supset(\wedge(a(), d()), \supset(W_H, \wedge(b(), c()))) \\
&\quad\mid \wedge(e(), e()) \\
c() &= \langle\mathrm{c}\rangle \\
d() &= \langle\mathrm{d}\rangle
\end{aligned}$$

with m, $\wedge$ defined as above.

3.4. Parallel communicating grammar systems

Cai *et al.* (2003) use parallel communicating grammar systems, or PCGSs (Păun and Sântean, 1990), for modeling pseudoknots. Like set-local multicomponent grammars, these work by parallel rewriting. However, a PCGS is not a grammar of multicomponent productions, but rather consists of $k + 1$ *component grammars* $\langle G_0, G_1, \ldots, G_k \rangle$. G_0 is called the *master*. Each grammar has, in addition to the normal nonterminal symbols, a set of *query symbols* $\{\mathrm{Q}_0, \ldots, \mathrm{Q}_\mathrm{k}\}$. A derivation begins with the $k + 1$ start symbols $\langle S_0, \ldots, S_k \rangle$ (as opposed to a multicomponent grammar, which would start with a single start symbol), and each grammar runs independently, except for the following:

- A query symbol Q_i can be rewritten with the current sentential form of G_i, and G_i is restarted at the next step.
- If a nonterminal symbol does not have any matching productions, it cannot be rewritten.
- A query symbol cannot be rewritten with a string containing another query symbol or a nonterminal symbol that cannot be rewritten.
- A nonterminal symbol cannot be rewritten if a rewritable query symbol is present.

Cai *et al.* use a restricted form of PCGS in which the auxiliary components are regular grammars (that is, CFGs whose right-hand sides have a single nonterminal, which are at the rightmost position). The master component looks like the following (simplified from the published version, which is itself simplified):

$$\mathrm{S} \rightarrow \mathrm{PQ_1PPXP}$$

$$\mathrm{X} \rightarrow \mathrm{aXu} \mid \mathrm{cXg} \mid \mathrm{gXc} \mid \mathrm{uXa}$$

$$\mathrm{X} \rightarrow \mathrm{PQ_2P}$$

$$\mathrm{P} \rightarrow \mathrm{aP} \mid \mathrm{cP} \mid \mathrm{gP} \mid \mathrm{uP} \mid \epsilon$$

$$\mathrm{H}_i \rightarrow \epsilon$$

And G_1, G_2, and G_3 are as follows (the vertical displacement of the columns is for clarity only):

$$\mathrm{S_1} \rightarrow \mathrm{Q_2}$$

$$\mathrm{T} \rightarrow \mathrm{T_1} \qquad \mathrm{S_2} \rightarrow \mathrm{T}$$

$$\mathrm{T_1} \rightarrow \mathrm{Q_3} \qquad \mathrm{T} \rightarrow \mathrm{Q_3}$$

$$\mathrm{A} \rightarrow \mathrm{aQ_3} \qquad \mathrm{A} \rightarrow \mathrm{Q_3u} \qquad \mathrm{S_3} \rightarrow \mathrm{A}$$

$$\mathrm{C} \rightarrow \mathrm{cQ_3} \qquad \mathrm{C} \rightarrow \mathrm{Q_3g} \qquad \mathrm{S_3} \rightarrow \mathrm{C}$$

$$\mathrm{G} \rightarrow \mathrm{gQ_3} \qquad \mathrm{G} \rightarrow \mathrm{Q_3c} \qquad \mathrm{S_3} \rightarrow \mathrm{G}$$

$$\mathrm{U} \rightarrow \mathrm{uQ_3} \qquad \mathrm{U} \rightarrow \mathrm{Q_3a} \qquad \mathrm{S_3} \rightarrow \mathrm{U}$$

$$\mathrm{H} \rightarrow \mathrm{H_1} \qquad \mathrm{H} \rightarrow \mathrm{H_2} \qquad \mathrm{S_3} \rightarrow \mathrm{H}$$

(The rules involving H_i are not in the original paper, but must be inserted in order for the derivation to proceed as shown in the paper. The distinction between $\mathrm{H_1}$ and $\mathrm{H_2}$ is needed to prevent G_1 from querying G_2 at the wrong time.)

G_1 and G_2 run in parallel to generate the two halves of a hairpin. Because of the linearity constraint, the sentential forms of these grammars always have exactly one nonterminal symbol, which was generated at the previous step. Therefore the operation of G_1 and G_2 resembles that of a set-local multicomponent grammar.

Furthermore, G_0, the pair $\langle G_1, G_2 \rangle$, and G_3 have time-independent derivations: G_1 halts when it generates the symbol $\mathrm{H_1}$, and only then can it be successfully queried by G_0. Similarly for G_2; and G_3 only

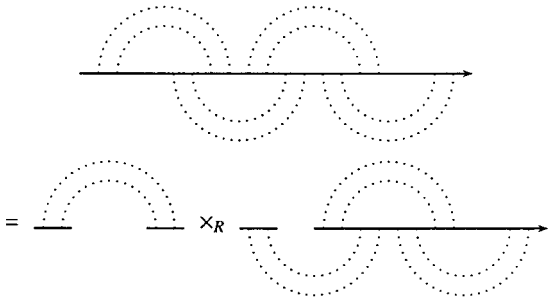

FIG. 9. "Multipseudoknot" generable by the grammar of Rivas and Eddy.

has derivations of length one. Therefore we can time-shift the component derivations without harm. This means that we can "inline" all the queries in a multicomponent grammar:[4]

$$\mathrm{S} \rightarrow \mathrm{PQ_1PPXP}$$

$$\langle \mathrm{Q_1, X} \rangle \rightarrow \langle \mathrm{Q_1, aXu} \rangle \mid \langle \mathrm{Q_1, cXg} \rangle \mid \langle \mathrm{Q_1, gXc} \rangle \mid \langle \mathrm{Q_1, uXa} \rangle$$

$$\langle \mathrm{Q_1, X} \rangle \rightarrow \langle \mathrm{Q_1, PQ_2P} \rangle$$

$$\langle \mathrm{Q_1, Q_2} \rangle \rightarrow \langle \mathrm{S_1, S_2} \rangle$$

$$\langle \mathrm{S_1, S_2} \rangle \rightarrow \langle \mathrm{Q_2, T} \rangle \rightarrow \langle \mathrm{T_2, S_2} \rangle \rightarrow \langle \mathrm{T_1, T} \rangle \rightarrow \langle \mathrm{Q_3, Q_3} \rangle$$

$$\langle \mathrm{Q_3, Q_3} \rangle \rightarrow \langle \mathrm{S_3, S_3} \rangle$$

$$\langle \mathrm{S_3, S_3} \rangle \rightarrow \langle \mathrm{A, A} \rangle \rightarrow \langle \mathrm{aQ_3, Q_3u} \rangle$$

$$\langle \mathrm{S_3, S_3} \rangle \rightarrow \langle \mathrm{C, C} \rangle \rightarrow \langle \mathrm{cQ_3, Q_3g} \rangle$$

$$\langle \mathrm{S_3, S_3} \rangle \rightarrow \langle \mathrm{G, G} \rangle \rightarrow \langle \mathrm{gQ_3, Q_3c} \rangle$$

$$\langle \mathrm{S_3, S_3} \rangle \rightarrow \langle \mathrm{U, U} \rangle \rightarrow \langle \mathrm{uQ_3, Q_3a} \rangle$$

$$\langle \mathrm{S_3, S_3} \rangle \rightarrow \langle \mathrm{H, H} \rangle \rightarrow \langle \mathrm{H_1, H_2} \rangle \rightarrow \langle \epsilon, \epsilon \rangle$$

And this grammar is not very different from Uemura's linear TAG. Since this grammar has rank two and fan-out two, it can be parsed in $\mathcal{O}(n^6)$ time, like Cai *et al.*'s. Further transformation could make the grammar linear, making it parseable in $\mathcal{O}(n^4)$ time like the simpler of Uemura's two grammars.

3.5. Regular-form TAG

We turn to another restriction on TAG, called *regular-form TAG* (Rogers, 1994), which generates the same languages as CFG and has the same parsing complexity as CFG of $\mathcal{O}(n^3)$. But it has greater derivational generative capacity than CFG (Chiang, 2002), which we use here for modeling limited RNA tertiary interactions.

In his original definition, the details of which we omit here, Rogers defines a restriction on TAG adjunction, called *regular adjunction*, that can generate only regular path sets. He then identifies the subclass of regular-form TAGs, which have the property that every derived tree that can be derived using unrestricted adjunction could also have been derived using only regular adjunction. But since Rogers'

[4]In the production $\langle \mathrm{Q_2, T} \rangle \rightarrow \langle \mathrm{T_2, S_2} \rangle$ we have made use of another trick, relying on the knowledge that the result of the query consists of a single nonterminal symbol.

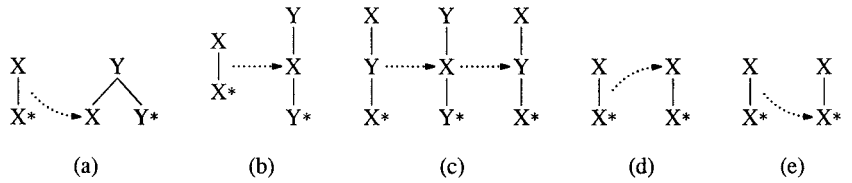

FIG. 10. Examples of adjunction in regular-form TAG. (**a**) Off-spine adjunction, allowed. (**b**) Acyclic spine adjunction, allowed. (**c**) Cyclic spine adjunction, not allowed. (**d**) Root adjunction, not allowed. (**e**) Foot adjunction, allowed.

recognition algorithm only performs regular adjunction, it cannot in general produce all possible derivations of a sentence and therefore cannot be used as a parser.

A more technical issue is that regular adjunction can occur at either the root or foot, which creates derivational ambiguity. Rogers' algorithm, however, cannot distinguish between the two. If we want the parser to compute derivations, one or the other should be disallowed. Following Schuler *et al.* (2000), we prohibit adjunction at the root. This leads us to the following definition, which narrows Rogers' definition to eliminate both of the above problems:

Definition 3. *We say that a TAG is in* regular form, *or an RF-TAG, if there exists some partial ordering* $\preceq$ *over nonterminal symbols such that if* β *is an auxiliary tree whose root and foot nodes are labeled* X*, and* η *is a node labeled* Y *on* β*'s spine where adjunction is allowed, then* $X \preceq Y$*, and* $X = Y$ *only if* η *is a foot node.*

Thus, adjunction at nodes not lying along the spine and adjunction at the foot node are allowed freely; adjunction at nodes lying along the spine is allowed to a bounded depth, but adjunction at the root is not allowed at all (Fig. 10).

Grammars of this type could be used for structures in which all but a bounded number of self-contacts are nested. For example, an RF-TAG can generate pseudoknots if there was a known upper bound on the number of self-contacts in one of the hairpins. A more appropriate application would be a cloverleaf structure (as in Fig. 1) in which two of the hairpins "kiss" (Fig. 4a), forming a small number of self-contacts crossing over an unbounded number of nested self-contacts. Such kissing hairpins within cloverleaf structures are known to occur in transfer RNAs. If the number of such self-contacts is indeed bounded, we can write an RF-TAG similar to the one above to generate them (Fig. 11).

3.6. Intersections of CFLs

Another strategy for obtaining more SGC out of a grammar formalism is to combine multiple grammars into a single system which accepts the intersection of the languages accepted by the component grammars, and which assigns to each string the *unification* (in some sense) of the structural descriptions assigned by the component grammars. This technique has not received much attention in computational linguistics,

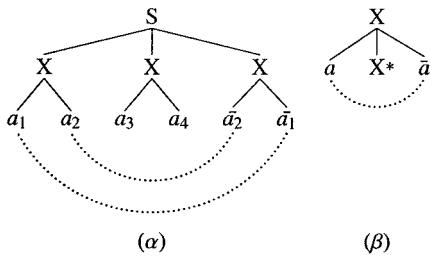

FIG. 11. RF-TAG for cloverleaf with kissing hairpins. The initial tree α generates the loops (here fixed to two monomers each), and β generates the stem regions.

probably because linguistic structures tend to be hierarchical, and it is not very clear how to unify multiple hierarchical structures into a single one. With molecular structures, on the other hand, it is straightforward to unify two linkings of a string, simply by merging them.

Context-free languages are not closed under intersection (Hopcroft and Ullman, 1979). This suggests the possibility of using two or more CFGs to recognize a language beyond the power of CFG, without going beyond its parsing complexity of $\mathcal{O}(n^3)$. Brown and Wilson (1996) propose just this approach for RNA pseudoknots. They observe that $\{a^m g^* u^m c^*\}$ and $\{a^* g^n u^* c^n\}$ are context-free languages, but their intersection is the non-context-free language $\{a^m g^n u^m c^n\}$. This language is reminiscent of a set of pseudoknots: the m a's and u's form one hairpin, and the n g's and c's are the other. Therefore this would seem to be an efficient way of modeling pseudoknots.

However, in order for the pseudoknot to be well-formed, the two hairpins must interlock without colliding. That is, the base pairings must cross, but no two pairings should involve the same base. But the only reason the above example achieves this is because one hairpin has only a's and u's and the other has only c's and g's—that is, each symbol indicates overtly which hairpin it belongs to. For real molecules, both component grammars would have to generate at least all possible hairpins. In that case, there would be no way of preventing the component grammars from missing each other or colliding.

The root of the problem is that intersection only operates on strings, not structural descriptions. It allows parallel structural descriptions to be derived independently, then filters them on the basis of their string yields. The above example attempts to harness this filtering to generate only well-formed pseudoknots, but in order to do so it assumes that there is more information in the string languages than there really is.

Brown and Wilson recognize that there is a difficulty, but they locate the difficulty in the calculation of probabilities using the model, rather than the model itself. Their solution is to employ a special parsing strategy that uses the results of parsing with the first grammar to constrain the parse with the second to ensure that the hairpins interlock without colliding; then the string is reparsed with the first, then again with the second. But since these parsing constraints are defined in terms of these particular grammars, it is not clear how their technique would be generalized to other pairs of grammars. Moreover, their method is by design only an approximation of the desired result.

We conclude that the single-grammar approaches are preferable to the intersected-CFL approach, despite having a higher asymptotic time complexity. See Section 4.2 below, however, for an example use of intersection *within* a single grammar.

4. PROTEIN SECONDARY STRUCTURE

Protein structure is in many ways more challenging to linguistic description than that of nucleic acids. We examine here two approaches to the specification of β-sheets: one using yet another equivalent of set-local multicomponent TAG; the other, a novel use of intersection in a *range concatenation grammar* (Groenink, 1997; Boullier, 2000). We then examine a pair of approaches for modeling interactions between α-helices: one using *multi-tape context-free grammars* (Lefebvre, 1996; Waldispühl and Steyaert, 2005), and one using ordinary context-free grammars (Chiang *et al.*, 2006).

4.1. Ranked node-rewriting grammars for β-sheets

Abe and Mamitsuka (1997) use a formalism called ranked node-rewriting grammar (RNRG) to generate β-sheets. RNRG is essentially TAG with multiple foot nodes on elementary trees. When a node η is rewritten with a tree β, the children of η are identified with the foot nodes of β, matched up according to linear precedence. RNRG(k) is the class of RNRGs which contain no elementary trees with more than k foot nodes. For an example of an RNRG(2), see Fig. 12.

Since rewriting with an elementary tree with one foot node corresponds to adjunction (and rewriting with an elementary tree with zero foot nodes corresponds to another standard operation, *substitution*, we do not make use of here), RNRG(1) is just a cosmetic variant of TAG (see Fig. 14).

A derived auxiliary tree with k foot nodes can be viewed as a tuple of $k+1$ strings, so that RNRG(k) can be thought of as an LCFRS with fan-out $k+1$. The rank of the LCFRS would be the maximum number of adjunction sites on any elementary tree. In order to simplify computation of inside and outside

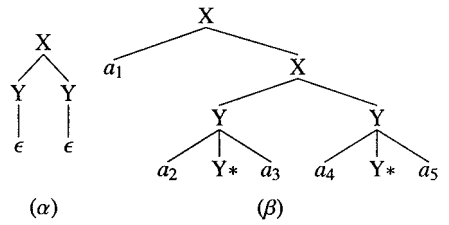

FIG. 12. RNRG for β-sheet of five strands.

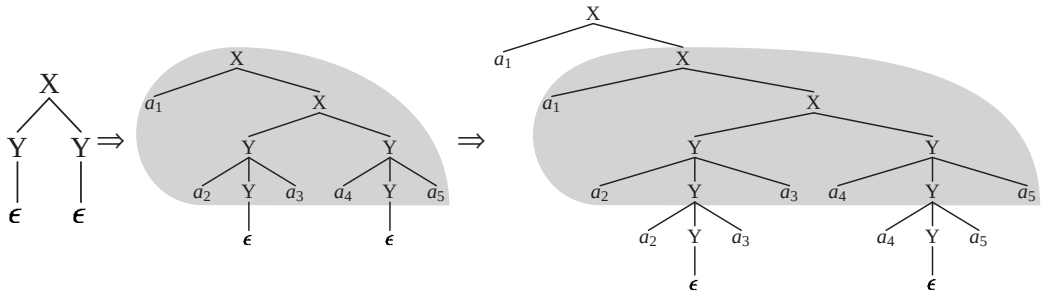

FIG. 13. Rewriting of α from Fig. 12 using two applications of β, with new material at each step shaded.

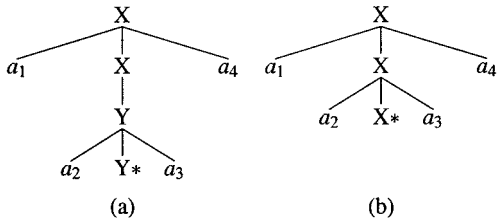

FIG. 14. RNRG(1) elementary tree and equivalent TAG elementary tree.

probabilities, Abe and Mamitsuka define linear RNRG(k), in which each auxiliary tree has at most one node at which rewriting (adjunction) is possible. Linear RNRG(k), because it is equivalent to LCFRS with rank one and fan-out $k+1$, has parsing complexity $\mathcal{O}(n^{2(k+1)})$.[5] They fix $k=1$, giving $\mathcal{O}(n^4)$. The other details need not concern us here.

Figure 12 shows a grammar for generating a β-sheet of five alternating strands; Fig. 13 shows an example derivation and Fig. 16 shows an example β-sheet. In Abe and Mamitsuka's analysis, the links are not explicitly drawn; the result is that each derivation corresponds to multiple β-sheet structures, each with a different spatial ordering of the strands. Thus, the derivations of the grammar of Fig. 14a would not be able to distinguish the β-sheets of Fig. 15.

Set-local multicomponent TAG offers a similar solution; Fig. 17 shows a grammar to generate the five-strand sheet of Fig. 2b. We can add links to the elementary trees (of either an RNRG or a multicomponent TAG) to distinguish permuted β-sheets (Fig. 16).

The difficulty is that parsing of these grammars is exponential in the number of strands per sheet. Moreover, every grammar imposes some upper bound, so that there is no single grammar that can generate all β-sheets. For this reason, approaches of this type appear to be prohibitively expensive.

[5] We conjecture that general RNRG(k) is not that much more difficult to parse, with a complexity of $\mathcal{O}(n^{2(k+1)+2})$, but we do not present an argument here.

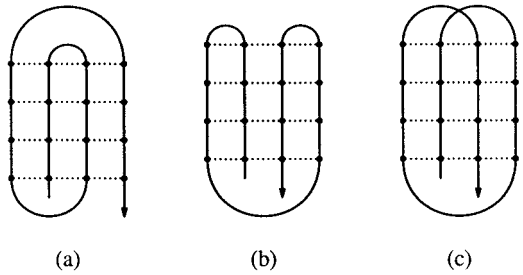

FIG. 15. Permuted β-sheets.

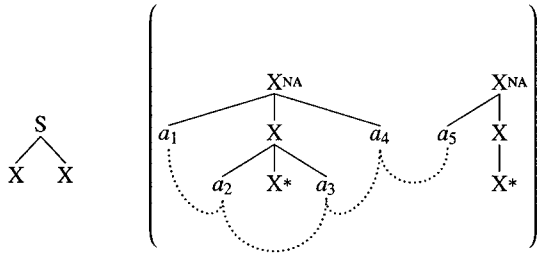

FIG. 16. Set-local multicomponent TAG for protein β-sheet of Fig. 2b.

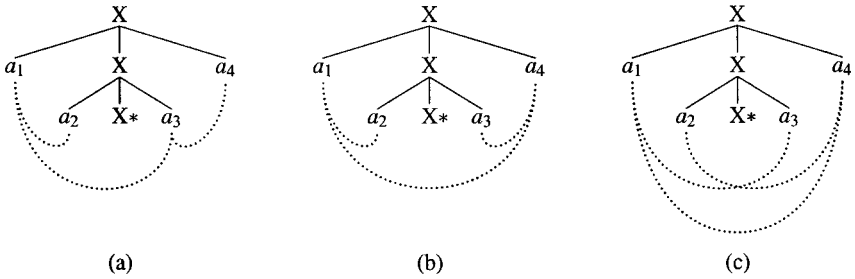

FIG. 17. Elementary trees for generating the permuted β-sheets of Fig. 15.

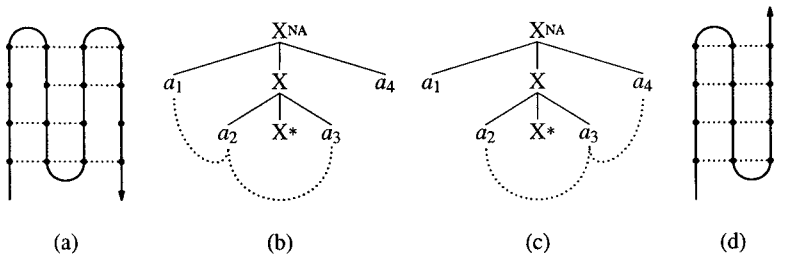

FIG. 18. Illustration of "spurious" structural ambiguity in a multicomponent TAG.

A second problem is that this analysis is susceptible to a kind of "spurious" structural ambiguity (Searls, 1999) in which a single structure can be derived in multiple ways (Fig. 18). In order to generate the β-sheet (a), we need trees like (b) and (c). But either of these trees can be used by itself to generate the β-sheet (d). The grammar must make room for the maximum number of strands, but when it does not use all of it, ambiguity can arise. It should be possible to carefully write the grammar to avoid much of this ambiguity, but we have not been able to eliminate all of it even for the single-component TAG case.

4.2. Range-concatenation grammars for β-sheets

Range concatenation grammars, or RCGs (Boullier, 2000), and the simple literal movement grammars (Groenink, 1997) on which they are based, are a class of grammars even larger than the LCFRSs. The class of languages they describe has a number of convenient properties: it is exactly the languages recognizable in deterministic polynomial time (Bertsch and Nederhof, 2001), and it is closed under intersection. The latter property makes RCGs potentially useful for modeling protein β-sheets in a way that does not suffer from the difficulties we pointed out with Brown and Wilson's intersection approach (Chiang, 2004).

We present here a brief definition of a variant of RCG as a kind of deductive system. RCG clauses have the form

$$\psi :- \phi_1, \ldots, \phi_n.$$

(meaning "ψ is provable if $\phi_1, \ldots, \phi_n$ all are"). If $n = 0$, we simply write

$$\psi.$$

(which is trivially provable). The ψ and the ϕ_i in turn have the form

$$A(\alpha_1, \ldots, \alpha_m)$$

where A is a predicate (nonterminal) symbol and the α_j are strings of terminal symbols and variables (which range over strings of terminal symbols). Every α_j in ψ must be a substring of an $\alpha_{j'}$ in one of the ϕ_i. This condition ensures that in the derivation of a string w, all variables are instantiated only to substrings of w. (The standard definition of RCG does not have this requirement, because its variables range not over strings but pairs of string positions of w. The definition here is closer to that of simple literal movement grammars [Groenink, 1997].) The language defined by an RCG is the set of all strings w such that $S(w)$ is provable, where S is a distinguished start predicate.

Moreover, since in an RCG there are no restrictions on what literals may be conjoined in the right-hand side of a production, RCG is closed under intersection: if Start_1 and Start_2 are the start predicates of two RCGs G_1 and G_2 (with disjoint nonterminal alphabets), create a new start predicate Start and add the production

$$\text{Start}(X) :- \text{Start}_1(X), \text{Start}_2(X)$$

which recognizes the intersection of the languages generated by G_1 and G_2. Thus RCG internalizes the intersection operation, which allows more control than Brown and Wilson's scheme. The caveats from our critique of that scheme still apply, however. For example, Boullier (1999) gives an RCG which he claims models German scrambling, a construction in which all the nouns of a sentence can appear in any order. His grammar checks for a verb for every noun and vice versa, using intersection to enforce all these constraints simultaneously. But like Brown and Wilson's system, it relies on some assumptions about the generated string to ensure that the constraints are properly coordinated.

Nevertheless, RCG's closure under intersection might be a useful property for modeling complex folds like protein β-sheets. We start with some building blocks:

$$\text{Anti}(a_1X, Ya_2) :- \text{Anti}(X, Y) \qquad a_i \in \Sigma$$

$$\text{Anti}(\epsilon, \epsilon)$$

$$\text{Par}(a_1X, a_2Y) :- \text{Par}(X, Y) \qquad a_i \in \Sigma$$

$$\text{Par}(\epsilon, \epsilon)$$

$$\text{Adj}(X, Y) :- \text{Ant}(X, Y)$$

$$\text{Adj}(X, Y) :- \text{Par}(X, Y)$$

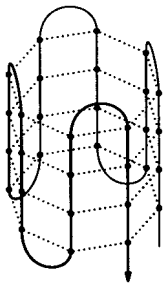

FIG. 19. β-Barrel.

The predicates Anti and Par generate pairs of adjacent antiparallel and parallel strands, respectively, and the predicate Adj generates two adjacent strands in either configuration. Irregularities as in Fig. 18a are also possible, but not shown here.

We can then use the intersection ability of RCG to combine these pairs of strands into a sheet. Thus the following grammar generates β-sheets where the strands are arranged according to their order in the sequence:

$$\text{Beta}(AB) \text{ :- } \text{B}(A, B)$$

$$\text{B}(ABY, B') \text{ :- } \text{B}(A, B), \text{Adj}(B, B')$$

$$\text{B}(BY, B') \text{ :- } \text{Adj}(B, B')$$

The first argument to B is a β-sheet minus the last strand, and the second argument is the last strand. The second production forms a larger β-sheet out of a smaller one by appending a new last strand and joining it to the previous last strand using Adj. This production has $\mathcal{O}(n^5)$ possible instantiations (because it takes six indices to specify the variables on the left-hand side, but the arguments of B are always adjacent, eliminating one index), and therefore the parsing complexity of this grammar is also $\mathcal{O}(n^5)$. Crucially, this complexity bound is not dependent on the number of strands, because each series of contacts is generated in sister subderivations, unlike the multicomponent TAG analysis.

But even sister subderivations can control each other via their root nonterminal (predicate) symbols, as illustrated in the following example. A β-sheet can be rolled into a cylinder to form a *β-barrel* (Fig. 19). We can generate these as well, but we must keep track of the direction of each strand, as in the grammar of Fig. 20, so as not to generate any Möbius strips. Here B has three arguments: the first strand, the middle part, and the last strand; there is an additional predicate symbol B′ which is the same as B, except that B′ is for sheets with antiparallel first and last strands, whereas B is restricted here to sheets with parallel first

$$\text{Barrel}(ABC) \text{ :- } \text{B}(A, B, C), \text{Par}(A, C)$$

$$\text{Barrel}(ABC) \text{ :- } \text{B}'(A, B, C), \text{Anti}(A, C)$$

$$\text{B}(A, BCY, C') \text{ :- } \text{B}'(A, B, C), \text{Anti}(C, C')$$

$$\text{B}(A, BCY, C') \text{ :- } \text{B}(A, B, C), \text{Par}(C, C')$$

$$\text{B}(A, Y, A') \text{ :- } \text{Par}(A, A')$$

$$\text{B}'(A, BCY, C') \text{ :- } \text{B}(A, B, C), \text{Anti}(C, C')$$

$$\text{B}'(A, BCY, C') \text{ :- } \text{B}'(A, B, C), \text{Par}(C, C')$$

$$\text{B}'(A, Y, A') \text{ :- } \text{Anti}(A, A')$$

FIG. 20. RCG for β-barrels.

and last strands. The first production joins the first and last strands to form a barrel; it uses the information in the B versus B′ distinction to join the strands so that no Möbius strips will be generated.

The strands of β-sheets do not always appear in linear order; they can be permuted as in Fig. 15. We can model such permutations by increasing the degree of synchronous parallelism (that is, the number of arguments to B), and therefore increasing parsing complexity. By contrast, since multicomponent TAG already uses synchronous parallelism to generate all the strands together, it allows permutations of strands at no extra cost.

Suppose we envision a sheet being built up one strand at a time, each successive strand being added to either side of the sheet:

$$\begin{aligned}
\text{Beta}(ABCD) &:- \text{B}(A, B, C, D)\\
\text{B}(ABC, D, Y, B') &:- \text{B}(A, B, C, D), \text{Adj}(B, B')\\
\text{B}(A, B, CDY, B') &:- \text{B}(A, B, C, D), \text{Adj}(D, B')\\
\text{B}(\epsilon, B, Y, B') &:- \text{Adj}(B, B')
\end{aligned}$$

Figure 15a shows an example sheet that can be generated by this grammar but not the previous ones. In this grammar, the second and fourth arguments to B are the leftmost and rightmost strands (*not* respectively) in the folded structure. The second production adds a new strand on one side, and the third production adds a new strand on the other. Both productions have $\mathcal{O}(n^7)$ possible instantiations if we take into account that the four arguments to B will always be adjacent.

Suppose we always build up a sheet out of two smaller sheets:

$$\begin{aligned}
\text{Beta}(ABCDE) &:- \text{B}(A, B, C, D, E)\\
\text{B}(ABC, D, EYA', B', C'D'E') &:- \text{B}(A, B, C, D, E), \text{B}(A', B', C', D', E'), \text{Adj}(B, D')\\
\text{B}(A, B, CDEYA', B', C', D', E') &:- \text{B}(A, B, C, D, E), \text{B}(A', B', C', D', E'), \text{Adj}(D, D')\\
\text{B}(\epsilon, B, C, D, \epsilon) &:- \text{Adj}(B, D)
\end{aligned}$$

Figure 15b shows an example sheet that can be generated by this grammar but not the previous ones. In this grammar, the second and fourth arguments are again the leftmost and rightmost strands (*not* respectively) in the folded structure. The second and third productions join two β-sheets together in two different ways; there are conceivably four ways to join them together, but using only these two avoids "spurious" structural ambiguity. Both productions have $\mathcal{O}(n^{12})$ possible instantiations if we take into account that the five arguments to B will always be adjacent.

Figure 15c shows the only permutation of four strands that the above grammar cannot generate. This may not be problematic, since, at least for sheets formed out of two hairpin motifs, we are not aware of instances of this permutation occurring in nature (Branden and Tooze, 1999).

It should be emphasized, however, that any energies or conformation counts added to these grammars will not be able to make the self-contacts between two strands dependent on self-contacts with other strands. Akutsu (2000) and Lyngsø and Pedersen (2000) have shown that certain formulations of the problem of predicting RNA secondary structures with generalized pseudoknots are NP-hard. Both of these proofs assume some kind of dependence between nonadjacent strands. Akutsu assumes that no residue can participate in two contacts (one on either side), which is true of RNA secondary structures but not of protein structures. Lyngsø and Pedersen assume that the energy of a base pairing (i, j) can be affected by another base pairing $(j-1, i')$ even if i and i' are in different strands (or by $(j', i+1)$ even if j and j' are in different strands); it remains to be seen whether such dependencies might be needed, for example, in calculating conformation counts for β-sheets.

4.3. α-helix bundles

Besides β-sheets, the other major class of protein secondary structure is the α-helix. As shown in Fig. 2a, the key intramolecular interactions in α-helices are local and cyclical, forming short crossing dependencies.

$$\text{Start}'(X) \text{ :– } \text{Start}(X, X)$$

$$\text{Start}(aX, Yb) \text{ :– } \text{Start}(X, Y)$$

$$\text{Start}(X, Y) \text{ :– } \text{Bundle}(X, Y)$$

$$\text{Bundle}(X_1X_2, Y_1Y_2) \text{ :– } \text{H}(X_1, Y_1), \text{Bundle}(X_2, Y_2)$$

$$\text{Bundle}(X, Y) \text{ :– } \text{H}(X, Y)$$

$$\text{H}(Xa, bY) \text{ :– } \text{H}(X, Y)$$

$$\text{H}(a, a)$$

FIG. 21. Simplified version of Waldispühl and Steyaert's grammar.

α-helices do not form long-distance dependencies, such as those between the strands of a β-sheet, as an aspect of their definition as a secondary structure. Outward-facing side-chains of α-helices certainly form interactions with other residues in a typical protein, but these are usually of a more irregular and singular nature and thus not well suited to generic description.

Major exceptions involve cases where α-helices contact each other in rough alignment and thus are able to interact along their lengths in some regular fashion. Examples include certain signalling proteins that occur in four-helix bundles, higher-order structures called coiled coils that are seen in structural proteins, and membrane proteins for which successive α-helices cross and recross the membrane. Often such helix bundles comprise successive α-helices in antiparallel relationship.

Waldispühl and Steyaert (2005) propose the use of *m-tape context-free grammars* (Lefebvre, 1996) for modeling pairings between α-helices. We can describe m-tape context-free grammars as RCGs in which every clause is of the form

$$A(\alpha_1, \ldots, \alpha_m) \text{ :– } B_1(x_{11}, \ldots, x_{1m}), \ldots B_n(x_{n1}, \ldots, x_{nm})$$

where the α_i are such that removing the terminal symbols from α_i yields $x_{1i} \cdots x_{ni}$. (Links can be specified between terminal symbols of a clause, but we omit a description here of how to restrict the links in order to reproduce the definition of m-tape CFGs.) The start predicate is an m-place predicate that defines a relation on m-tuples of strings. With respect to each of the m arguments, the grammar behaves like an ordinary CFG because of the above restriction.

In Waldispühl and Steyaert's system, $m = 2$, and the two strings are constrained to be equal. We can formalize this constraint within an RCG by adding a new start predicate Start$'$ defined by

$$\text{Start}'(X) \text{ :– } \text{Start}(X, X)$$

This gives the grammar two copies of the string to work with, so that self-contacts can be modeled as contacts from one copy to the other. See Fig. 21 for a highly simplified version of Waldispühl and Steyaert's grammar, and Fig. 22 for an example structure.

This superposition of the two sides of the grammar is reminiscent of intersection; indeed, Lefebvre notes the connection between his method and Brown and Wilson's intersected-CFL model of RNA pseudoknots.

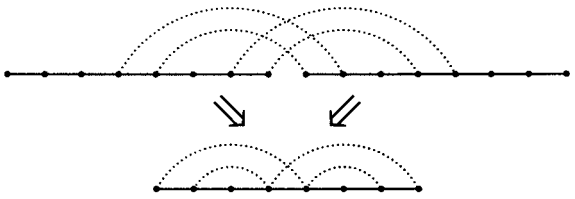

FIG. 22. Example structure generated by the grammar of Fig. 21. (**Above**) Two copies viewed separately. (**Below**) Two copies merged.

And as in Brown and Wilson's system, there is not, as far as we can tell, a mechanism for ensuring that the two sides of the grammar agree on the extent of each helix.

Elsewhere (Chiang *et al.*, 2006), we present a less ambitious model of helix bundles, one which uses a context-free grammar based on the model of Chen and Dill (1995, 1998) (see also Mauri *et al.*, 1999) intersected with a finite-state model of α-helices based on the Zimm-Bragg model (Zimm and Bragg, 1959) to model two-helix bundles. Formally, it does not go beyond the power of context-free grammars and is therefore less general than Waldispühl and Steyaert's approach. However, it attempts to model the full energy landscape of the chain and not just the minimum- or near-minimum energy structures. Moreover, this model could perhaps be extended along the lines of Abe and Mamitsuka's approach to β-sheets or our RCG-based account (Section 4.2) to yield a much more general model of helix bundles. This is a subject for future investigation.

5. CONCLUSION

Formal grammars provide an abstract view of RNA/protein structure, one that should enable both high-level descriptions and efficient searching through large spaces of structures; the key element of this view is that self-contacts in a chain are localized to elementary structures that combine with each other by grammar operations that are independent of the global context. All the approaches we have surveyed here share this principle in common with Searls' original work, and we have shown how all but two are equivalent or very similar to TAG or multicomponent TAG. The two exceptions, Brown and Wilson's intersected-CFL method and Waldispühl and Steyaert's two-tape CFG method, can be captured in the more general RCG formalism.

We point out two possible new directions. First, it is important to distinguish between the WGC and SGC of a grammar formalism. It would be desirable to increase the SGC of a formalism with respect to structures without increasing its WGC or its processing complexity. Our RF-TAG analysis of limited RNA tertiary structure is an example of such an approach. Second, intersection has not found many applications in computational linguistics, whose mainly hierarchical structures cannot be superimposed easily. But they may prove to be useful for modeling biological structures such as β-sheets—provided they are used properly, in a system like RCG.

ACKNOWLEDGMENTS

We would like to thank Adam Lucas, Julia Hockenmaier, Liang Huang, and the anonymous reviewer for their helpful comments. This work was partially supported by NSF-ITR grant EIA02-05456.

REFERENCES

Abe, N., and Mamitsuka, H. 1997. Predicting protein secondary structures based on stochastic tree grammars. *Machine Learn.* 29, 275–301.

Akutsu, T. 2000. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Appl. Math.* 104, 45–62.

Becker, T., Joshi, A., and Rambow, O. 1991. Long distance scrambling and tree adjoining grammar. *Proc. 5th Conf. Eur. Chapter Assoc. Comput. Linguistics (EACL)*, 21–26.

Bertsch, E., and Nederhof, M.-J. 2001. On the complexity of some extensions of RCG parsing. *Proc. 7th Int. Workshop Parsing Technol. (IWPT)*, 66–77.

Berwick, R.C. 1984. Strong generative capacity, weak generative capacity, and modern linguistic theories. *Comput. Linguistics* 10, 189–202.

Boullier, P. 1999. Chinese numbers, MIX, scrambling, and range concatenation grammars. *Proc. 9th Conf. Eur. Chapter Assoc. Comput. Linguistics (EACL)*, 53–60.

Boullier, P. 2000. Range concatenation grammars. *Proc. 6th Int. Workshop Parsing Technol. (IWPT)*, 53–64.

Branden, C.-I., and Tooze, J. 1999. *Introduction to Protein Structure*, 2nd ed., Garland, New York.

Brown, M., and Wilson, C. 1996. RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search. *Proc. Pac. Symp. Biocomput.*, 109–125.

Burge, C., and Karlin, S. 1997. Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.* 268, 78–94.

Cai, L., Malmberg, R.L., and Wu, Y. 2003. Stochastic modeling of RNA pseudoknotted structures: A grammatical approach. *Bioinformatics* 19, i66–i73.

Chen, S.-J., and Dill, K.A. 1995. Statistical thermodynamics of double-stranded polymer molecules. *J. Chem. Phys.* 103, 5802–5813.

Chen, S.-J., and Dill, K.A. 1998. Theory for the conformational changes of double-stranded chain molecules. *J. Chem. Phys.* 109, 4602–4616.

Chiang, D. 2002. Putting some weakly context-free formalisms in order. *Proc. 6th Int. Workshop TAG Rel. Formalisms (TAG+)*, 11–18.

Chiang, D. 2004. Uses and abuses of intersected languages. *Proc. 7th Int. Workshop TAG Rel. Formalisms (TAG+)*, 9–15.

Chiang, D., and Joshi, A.K. 2002. Formal grammars for estimating partition functions of double-stranded chain molecules. *Proc. 2nd Int. Conf. Human Lang. Technol. Res. (HLT)*, 63–67.

Chiang, D., Joshi, A.K., and Dill, K.A. 2006. A grammatical theory for the conformational changes of simple helix bundles. *J. Comput. Biol.* 13, 21–42.

Dong, S., and Searls, D.B. 1994. Gene structure prediction by linguistic methods. *Genomics* 23, 540–551.

Durbin, R., Eddy, S.R., Krogh, A., *et al.* 1999. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, Cambridge, UK.

Groenink, A.V. 1997. *Surface without Structure: Word Order and Tractability Issues in Natural Language Analysis*. PhD thesis, University of Utrecht.

Hilbers, C.W., Michiels, P.J.A., and Heus, H.A. 1998. New developments in structure determination of pseudoknots. *Biopolymers* 48, 137–153.

Hopcroft, J.E., and Ullman, J.D. 1979. *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA.

Joshi, A.K. 1985. Tree adjoining grammars: How much context-sensitivity is necessary for assigning structural descriptions? *in* Dowty, D., Karttunen, L., and Zwicky, A., eds., *Natural Language Parsing*, 206–250, Cambridge University Press, Cambridge, UK.

Joshi, A.K. 2004. Starting with complex primitives pays off: Complicate locally, simplify globally. *Cogn. Sci.* 28, 637–668.

Joshi, A.K., Levy, L., and Takahashi, M. 1975. Tree adjunct grammars. *J. Comput. Syst. Sci.* 10, 136–163.

Joshi, A.K., and Schabes, Y., 1997. Tree-adjoining grammars, *in* Rosenberg, G., and Salomaa, A., eds., *Handbook of Formal Languages and Automata*, Volume 3, 69–124, Springer-Verlag, Heidelberg.

Kato, Y., Seki, H., and Kasami, T. 2004. Subclasses of tree adjoining grammar for RNA secondary structure. *Proc. 7th Int. Workshop TAG Rel. Formalisms (TAG+)*, 48–55.

Lefebvre, F. 1996. A grammar-based unification of several alignment and folding algorithms. *Proc. Int. Conf. Intellig. Syst. Mol. Biol.*, 143–154.

Lyngsø, R.B., and Pedersen, C.N.S. 2000. RNA pseudoknot prediction in energy-based models. *J. Comput. Biol.* 7, 409–427.

Mauri, G., Piccolboni, A., and Pavesi, G. 1999. Approximation algorithms for protein folding prediction (short). *Proc. 10th Ann. Symp. Discrete Algorithms*, 945–946.

Mel'čuk, I.A. 1988. *Dependency Syntax: Theory and Practice*, SUNY Press, Albany, NY.

Nebel, M.E. 2002. Combinatorial properties of RNA secondary structures. *J. Comput. Biol.* 9, 541–573.

Păun, G., and Sântean, L. 1990. Further remarks on parallel communicating grammar systems. *Int. J. Comput. Math.* 34, 187–203.

Rambow, O., and Satta, G. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theor. Comput. Sci.* 223, 87–120.

Reese, M.G., Kulp, D., Tammana, H., *et al.* 2000. `Genie`—gene finding in *Drosophila melanogaster*. *Genome Res.* 10, 529–538.

Rivas, E., and Eddy, S.R. 2000. The language of RNA: A formal grammar that includes pseudoknots. *Bioinformatics* 16, 334–340.

Rogers, J. 1994. Capturing CFLs with tree adjoining grammars. *Proc. 32nd Annu. Mtg. ACL*, 155–162.

Rogers, J. 2003. Syntactic structures as multi-dimensional trees. *Res. Language Comput.* 1, 265–305.

Sakakibara, Y., Brown, M., Hughey, R., *et al.* 1994. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Res.* 22, 5112–5120.

Satta, G., and Schuler, W. 1998. Restrictions on tree adjoining languages. *Proc. COLING-ACL*, 1176–1182.

Schuler, W., Chiang, D., and Dras, M. 2000. Multi-component TAG and notions of formal power. *Proc. 38th Annu. Mtg. ACL*, 448–455.

Searls, D.B. 1988. Representing genetic information with formal grammars. *Proc. 7th Natl. Conf. Artif. Intellig. (AAAI)*, 386–391.

Searls, D.B. 1992. The linguistics of DNA. *Am. Sci.* 80, 579–591.

Searls, D.B. 1999. Formal language theory and biological macromolecules, *in* Farach-Colton, M., Roberts, F.S., Vingron, M., *et al.*, eds., *Mathematical Support for Molecular Biology*, 117–140, American Mathematical Society, Providence, RI.

Uemura, Y., Hasegawa, A., Kobayashi, S., *et al.* 1999. Tree adjoining grammars for RNA structure prediction. *Theor. Comput. Sci.* 210, 277–303.

Viennot, G., and de Chaumont, M.V. 1983. Enumeration of RNA secondary structures by complexity, *in* Capasso, V., Grosso, E., and Paveri-Fontana, S.L., eds., *Mathematics in Biology and Medicine, Number 57 in Lecture Notes in Bbiomathematics*, 360–365. Conference proceedings: Berlin, New York: Springer-Verlag.

Vijay-Shanker, K., Weir, D., and Joshi, A. 1987. Characterizing structural descriptions produced by various grammatical formalisms. *Proc. 25th Annu. Mtg. ACL*, 104–111.

Waldispühl, J., and Steyaert, J.-M. 2005. Modeling and predicting all-α transmembrane proteins including helix-helix pairing. *Theor. Comput. Sci.* 335, 67–92.

Weir, D.J. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*, PhD thesis, University of Pennsylvania, Philadelphia.

Westhead, D.R., Slidel, T.W., Flores, T.P., *et al.* 1999. Protein structural topology: Automated analysis and diagrammatic representation. *Protein Sci.* 8, 897–904.

Zimm, B.H., and Bragg, J.K. 1959. Theory of the phase transition between helix and random coil in polypeptide chains. *J. Chem. Phys.* 31, 526–535.

Address correspondence to:
David Chiang
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292

E-mail: chiang@isi.edu