Formal Language Theory and Biological Macromolecules

David B. Searls
Bioinformatics Group
SmithKline Beecham Pharmaceuticals

Abstract

Biological macromolecules can be viewed, at one level, as strings of symbols. Collections of such molecules can thus be considered to be sets of strings, i.e. formal languages. This article reviews language-theoretic approaches to describing intramolecular and intermolecular structural interactions within these molecules, and evolutionary relationships between them.

1 Introduction

The author has for some time been investigating the application of formal language theory to biological macromolecules, primarily nucleic acids because of the relative simplicity of the biochemical structures and interactions. After introducing the very simple mathematical foundations for these investigations, this article will review three major lines of research. These can largely be found in more fully developed form in referenced publications, though some new material is also included in each case. The sections below will deal with the use of formal grammars to describe *intramolecular* interactions [17, 18, 21], a new class of grammars designed to encompass *intermolecular* interactions in assemblages of macromolecules [22], and automata-theoretic approaches to the alignment of lexical strings, and thus their *evolutionary* structure with respect to each other [18, 24].

The basis for molecular interactions within and between nucleic acid molecules is the *complementarity* of the four-letter alphabet. We generalize this as follows:

Definition 1.1 (Complementarity) *A* complementary alphabet *is a pair consisting of an alphabet* Σ *and a bijection* $c : \Sigma \rightarrow \Sigma$ *such that for all* $b \in \Sigma$ *there is a* $d \in \Sigma$, $b \neq d$, *for which* $c(b) = d$ *and* $c(d) = b$.

We will simply use Σ to denote such a complementary alphabet, and the bar notation $\overline{b}$ for $c(b)$. Note that the alphabet of DNA, $\Sigma_{\text{DNA}} = \{\, \text{a}, \text{c}, \text{g}, \text{t} \,\}$, can be seen to complementary, where $\overline{\text{a}} = \text{t}$, $\overline{\text{c}} = \text{g}$, etc. For purposes of modelling DNA the following simple properties of complementarity and string reversal are noted:

Lemma 1.1 (Reverse Complementarity) *For a complementary alphabet* Σ *and any* $u, v, w \in \Sigma^*$,

1. $\overline{(uv)} = \overline{u} \cdot \overline{v}$ *(i.e. complementarity extends to a string homomorphism)*
2. $\overline{(\overline{w})} = w$ *and* $(w^R)^R = w$
3. $\overline{(w^R)} = (\overline{w})^R$ *(which is thus simply written* $\overline{w}^R$*)*
4. $\overline{(uv)}^R = \overline{v}^R \cdot \overline{u}^R$

Proof: By straightforward inductions. □

For a string $w \in \Sigma^*_{\text{DNA}}$ of DNA, the reverse complement $\overline{w}^R$ can be seen to model the *opposite strand* of w, since in fact the two strands of the double helix are both complementary and antiparallel in orientation. Reverse complementarity can also be viewed as an operation on strings that corresponds to the process of DNA *replication*, since a new strand of DNA is laid down in the direction opposite to that of the template, taking the complement of each base. This dual view of reverse complementarity derives from the following result which, though mathematically trivial, can be seen as the "fundamental theorem" of the structure of DNA:

Theorem 1.1 (Watson & Crick, 1953) *Replication of the opposite strand of a string of DNA yields back the original string.*
Proof: Watson and Crick's famous understatement was "It has not escaped our attention that the specific pairing we have postulated immediately suggests a possible copying mechanism for the genetic material" [28]. In fact, for *any* complementary alphabet Σ and any $w \in \Sigma^*$, it follows immediately from Lemma 1.1 that $\overline{(\overline{w}^R)}^R = \overline{((\overline{w}^R)^R)} = \overline{(\overline{w})} = w$. $\square$

Viewing a double-stranded DNA molecule as a set $D = \{ w, \overline{w}^R \}$, it is interesting to see how some classic experimental observations served to contribute to this understanding of the nature of DNA. We use the notation $|\, w \,|_u$ to denote the number of occurrences of the substring u in the string $w \in \Sigma^*$, and for a set of strings S we let $|\, S \,|_u = \sum_{w \in S} |\, w \,|_u$.

Lemma 1.2 (Substring Complementarity) *For a complementary alphabet Σ and any* complementary pair *$D = \{ w, \overline{w}^R \}$ where $w \in \Sigma^*$, it is the case that for all $u \in \Sigma^+$, $|\, D \,|_u = |\, D \,|_{\overline{u}^R}$.*
Proof: For every instance of u in w, $\overline{u}^R$ will occur in $\overline{w}^R$, and vice versa. $\square$

The following empirical observations correspond to the two simplest cases of this lemma:

Corollary 1.1 (Chargaff, 1950) *In double-stranded DNA, the ratio of* 'a'*s to* 't'*s is one, as is the ratio of* 'g'*s to* 'c'*s.*
Proof: For a complementary pair D over alphabet Σ, $|\, D \,|_b = |\, D \,|_{\overline{b}}$ for all $b \in \Sigma$, from Lemma 1.2. Chargaff's observation [2] was evidence used by Watson and Crick to support the notion of base complementarity in their structure, though by itself it says nothing about directionality. $\square$

Corollary 1.2 (Kornberg, 1961) *In double-stranded DNA, the frequency with which a given base is the nearest neighbor of another in a specified direction, is the same as the frequency with which the complement of the second base is the nearest neighbor of the complement of the first.*
Proof: For a complementary pair D over alphabet Σ, $|\, D \,|_{bd} = |\, D \,|_{\overline{db}}$ for all $b, d \in \Sigma$, from Lemma 1.2. Kornberg and colleagues used a trick to move a radioactive label from a given type of base to its nearest neighbors, and by counting those neighbors confirmed the antiparallel directionality of the DNA strands and their replication [13]. $\square$

One final observation is offered that will prove to be crucial to the structure of nucleic acids; it deals with the special case of a complementary pair[1] that is in fact a singleton set:

Lemma 1.3 (Dyad Symmetry) *For any* $w \in \Sigma_{\text{DNA}}^*$, $w = \overline{w}^R$ *iff* $w = u\overline{u}^R$ *for some* $u \in \Sigma_{\text{DNA}}^*$.
Proof: Using Lemma 1.1:

(If:) Whenever $w = u\overline{u}^R$, we have $\overline{w}^R = \overline{(u\overline{u}^R)}^R = \overline{(\overline{u}^R)}^R \cdot \overline{u}^R = u\overline{u}^R = w$.

(Only if:) Note that w must be of even length, else it has a centermost $b \in \Sigma_{\text{DNA}}$ such that $b = \overline{b}$, which is disallowed by Definition 1.1. Thus we can divide any such w into equal halves u, v and we see that $w = uv = \overline{w}^R = \overline{v}^R\overline{u}^R = u\overline{u}^R$. □

That is, for a complementary pair in which the opposite strands are identical to each other, each of those strands can be formed by concatenating two strands from another complementary pair. Such concatenated complementary pairs will prove to be important in the following section.

2 Intramolecular Structure

Consider the set of DNA strings that exhibit dyad symmetry, as introduced in Lemma 1.3:

$$L_h = \{\, w \in \Sigma_{\text{DNA}}^* \mid w = \overline{w}^R \,\} = \{\, u\overline{u}^R \mid u \in \Sigma_{\text{DNA}}^* \} \tag{1}$$

We have seen that this language, L_h, consists of all strings formed by concatenating a complementary pair. Since the DNA double helix consists of antiparallel strands, we can imagine those strands to be connected head-to-tail at one end of a double helix. This in fact would constitute what is called a *hairpin* molecule: a single strand that folds back on itself and base-pairs in the same fashion as the two distinct strands of a double helix would.

The form of L_h immediately suggests a palindromic language, and in fact $L_h = L(G_h)$ for the simple context-free grammar

$$G_h: \quad S \to bS\overline{b} \mid \epsilon \quad \text{where} \;\; b \in \Sigma_{\text{DNA}} \tag{2}$$

[1] Note that there is little reason to consider a complementary pair to be ordered, given the obvious symmetry.

L_h is thus a context-free language, and it is also easily shown not to be regular. Does this imply that the language of DNA is not regular?

To make such an assertion would require a formal statement of what is meant by the "language of DNA". This has been discussed at length previously [18], but for purposes of this review we adopt the expedient of considering the language of DNA to be loosely specified by a series of biological phenomena which are deemed to be important *in vivo*, and whose manifestations can be linguistically formalized, at least in idealized form.

The grammar G_h is idealized in several senses. First, base-paired regions or *stems* need not be perfectly base-paired to form actual structures *in vivo*; in fact, in RNA where such folded structures are more common, not only are occasional mismatches tolerated but certain other non-complementary pairings are found with an intermediate degree of preference[2]. RNA folding is, overall, more a matter of thermodynamics than discrete mathematics. Second, the hairpin structure in particular is unrealistic because nucleic acids cannot "turn on a dime", and in reality *steric hindrance* would restrict the radius of curvature at the turn so that at least three bases would be unpaired. A more realistic grammar would be one that explicitly recognizes the potential for an unpaired stretch of bases at the turn, forming what is called a *stem-and-loop* structure:

$$G_{sl}: \quad S \to bS\bar{b} \mid A \qquad\qquad A \to bA \mid \varepsilon \tag{3}$$

However, $L(G_{sl}) = \Sigma^*$, so that the formal stem-and-loop grammar is useless with respect to its *weak generative capacity*, that is, the set of strings it generates. Of greater interest to computational linguists, however, are the *structural* descriptions of strings that are implicit in their derivations from a given grammar. These are typically portrayed by means of derivation trees, as in Figure 1.

The derivation tree shown for the hairpin grammar is particularly noteworthy in one respect: the appearance of the tree closely matches the actual physical structure of the folded molecule *in vivo* as it is usually portrayed schematically. Fundamentally, to a computational linguist, grammars capture systematic *dependencies* between symbols in a string, and it is considered

[2]In RNA, the base 'u' substitutes for the base 't', and 'g'/'u' pairs are observed. We will generally continue to use Σ_{DNA} even in cases where RNA is understood, because of the obvious bijection and because the RNA is encoded in DNA originally.

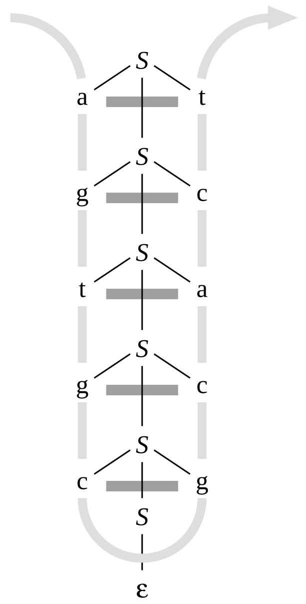

Figure 1: A hairpin derivation tree.

a virtue of a grammar formalism when the dependencies are both encoded in the individual productions of the grammar, and graphically preserved in the resulting derivation trees. Here is a case where the S rule of G_h embodies a single base pair, and the derivation tree in fact literally draws the secondary structure.

These virtues are largely lost in the stem-and-loop grammar, since strings that would be capable of forming stems might just as easily be generated by the loop portion of the grammar, i.e. the A rule in G_{sl}. We can recover some of the functionality of the grammar by way of versions like the following:

$$S \to bS\overline{b} \mid bAd \mid b \mid \epsilon \quad \text{where } b \neq \overline{d} \qquad A \to Ab \mid \epsilon \tag{4}$$

While this grammar can still generate strings that do not base-pair, it at least has the property that any stems that *can* be formed must be generated via the S rule and thus structurally captured by the grammar. The author has previously described other versions of stem-and-loop grammars that impose different constraints on the stems and loops [17, 18], but these complications are of little theoretical interest and we have found it appropriate instead to deal with the ideal case of complete base-pairing. This is formalized as follows:

Definition 2.1 (Ideality) *A string w over a complementary alphabet Σ is called* ideal *iff* $|w|_b = |w|_{\overline{b}}$ *for all $b \in \Sigma$. A language is ideal iff it contains only ideal strings.*

Intuitively, the point of an ideal string of DNA is that it at least theoretically has the potential to be completely base-paired. Thus it can be seen that the hairpin language L_h is an ideal form of the stem-and-loop language L_{sl}.

The hairpin grammar G_h is *linear* (i.e. no more than one nonterminal appears on the right-hand side of any rule), so that the folded *secondary structure* of the molecules depicted (as opposed their lexical sequence or *primary structure*) never branches. Yet branched secondary structure is a common theme in RNA folding, with new stems "budding" off the sides of other stems. We can formalize this type of secondary structure with an inductive definition:

Definition 2.2 (Orthodoxy) *A string w over a complementary alphabet Σ is called* orthodox *iff it is (1) the empty string ϵ, or (2) the result of inserting two adjacent complementary elements $b\overline{b}$, for some $b \in \Sigma$, anywhere in an orthodox string. A language is orthodox iff it contains only orthodox strings.*

The intuition behind this definition is that a new $b\overline{b}$ can either be placed at the end of an existing stem to grow it further, or in the side of one to start a new branch, and these operations suffice to form arbitrary such structures. We then observe the following relationship:

Lemma 2.1 (Ideality and Orthodoxy) *For a complementary alphabet Σ, the orthodox strings are a proper subset of the ideal strings if $|\Sigma| > 2$, and are equivalent to them otherwise.*
Proof: First note that any orthodox string on any sized alphabet is ideal, since only a pair of complementary elements are ever added in Definition 2.2. For Σ_{DNA}, the string 'actg' is ideal but not orthodox, but for an alphabet of two characters (e.g. 'g' and 'c') any ideal string is orthodox. This is most easily seen by observing that for any string with an equal number of 'g's and

'c's, there must be at least one place where a 'g' is found next to a 'c'; this pair can be removed, and the string must still be ideal. This process can thus be iterated to arrive at the empty string, and obviously it can also be reversed to produce the original string, which must therefore be orthodox. □

This observation suggests one reason why the alphabet of nucleic acids needs to be four bases instead of only two: in the latter case, orthodox secondary structure would be unavoidable and it might transpire that too much of it could interfere with the other business of the RNA molecules.

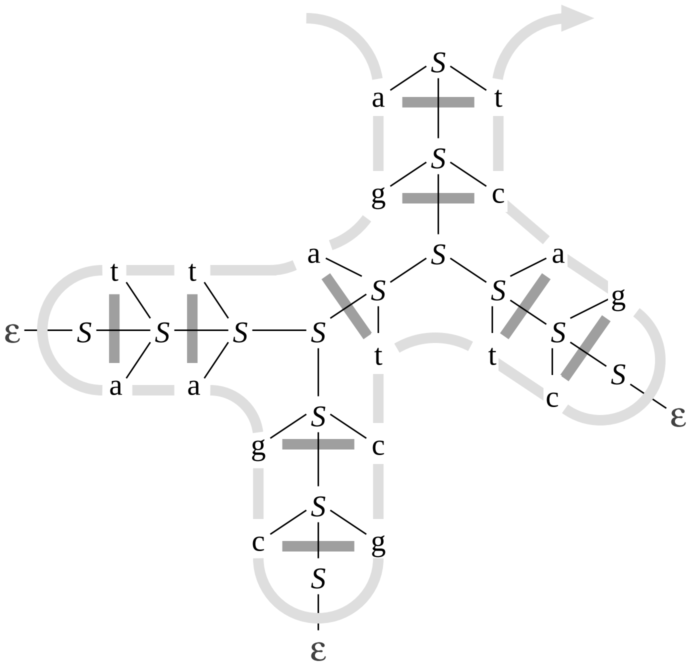

Figure 2: A branched derivation tree.

Consider the language L_o specified by the following grammar:

$$G_o: \quad S \rightarrow bS\overline{b} \mid SS \mid \epsilon \tag{5}$$

That L_o is simply the set of all orthodox strings is easily demonstrated (see [18], pp. 74–75). Figure 2 illustrates the sort of branched secondary structure that is captured by this grammar. Recently a number of workers have taken advantage of this characterization to develop stochastic forms of such grammars that are useful in recognizing certain orthodox secondary structures [8, 14, 15].

We have also noted previously that, while L_h is clearly nondeterministic (it being necessary to guess at the midpoint of the hairpin), L_o is, surprisingly, deterministic [18]. We now offer the following Griebach normal form grammar that is weakly equivalent to G_o, but deterministic:

$$\begin{array}{lll} G_{od}: & S \to bS_bS \mid \epsilon & \text{for each } b \in \Sigma_{\mathrm{DNA}} \\ & S_b \to \overline{b} \mid dS_dS_b & \text{for } d \neq \overline{b} \end{array} \tag{6}$$

Thus, the most general language of orthodox secondary structure is deterministic, while many useful subclasses of orthodox structure are not, including the hairpin language L_h and the classic *dumbbell* language of adjacent stems, $L_d = \{u\overline{u}^R v\overline{v}^R \mid u, v \in \Sigma^*_{\mathrm{DNA}}\}$, specified by

$$G_d: \quad S \to AA \qquad A \to bA\overline{b} \mid \epsilon \tag{7}$$

However, there is an important sense in which the deterministic grammar G_{od} fails to capture the structural aspects of the domain. Being deterministic, it is obviously unambiguous, i.e. there is a single leftmost derivation for any string in the language. On the other hand, G_o is ambiguous, as shown for example in the following four derivations for the same terminal string, 'gatcgatc':

$$\begin{array}{l} S \Rightarrow \mathrm{g}S\mathrm{c} \Rightarrow \mathrm{ga}S\mathrm{tc} \Rightarrow \mathrm{gat}S\mathrm{atc} \Rightarrow \mathrm{gatc}S\mathrm{gatc} \Rightarrow \mathrm{gatcgatc} \\ S \Rightarrow SS \Rightarrow \mathrm{g}S\mathrm{c}S \Rightarrow \mathrm{ga}S\mathrm{tc}S \Rightarrow \mathrm{gatc}S \Rightarrow \mathrm{gatcg}S\mathrm{c} \\ \qquad \Rightarrow \mathrm{gatcga}S\mathrm{tc} \Rightarrow \mathrm{gatcgatc} \\ S \Rightarrow \mathrm{g}S\mathrm{c} \Rightarrow \mathrm{g}SS\mathrm{c} \Rightarrow \mathrm{g}SSS\mathrm{c} \Rightarrow \mathrm{ga}S\mathrm{t}SS\mathrm{c} \Rightarrow \mathrm{gat}SS\mathrm{c} \\ \qquad \Rightarrow \mathrm{gatc}S\mathrm{g}S\mathrm{c} \Rightarrow \mathrm{gatcg}S\mathrm{c} \Rightarrow \mathrm{gatcga}S\mathrm{tc} \Rightarrow \mathrm{gatcgatc} \\ S \Rightarrow \mathrm{g}S\mathrm{c} \Rightarrow \mathrm{g}SS\mathrm{c} \Rightarrow \mathrm{ga}S\mathrm{t}S\mathrm{c} \Rightarrow \mathrm{gat}S\mathrm{c} \Rightarrow \mathrm{gat}SS\mathrm{c} \\ \qquad \Rightarrow \mathrm{gatc}S\mathrm{g}S\mathrm{c} \Rightarrow \mathrm{gatcg}S\mathrm{c} \Rightarrow \mathrm{gatcga}S\mathrm{tc} \Rightarrow \mathrm{gatcgatc} \end{array} \tag{8}$$

Note that the first derivation in (8) corresponds to a simple hairpin, and could just as easily have been derived from G_h. Similarly, the second derivation follows the pattern of the dumbbell grammar G_d. The last two form a new pattern, and if the derivation trees are drawn they can be seen to form the same cruciform structure [18], an instance of a class of *cloverleaf* structures $L_{c_n} = \{uv_1\overline{v}_1^R v_2\overline{v}_2^R \cdots v_n\overline{v}_n^R\overline{u}^R \mid u, v_1, v_2, \cdots v_n \in \Sigma_{\mathrm{DNA}}^*, n \geq 0\}$, typified by transfer RNA molecules ($n = 3$) and captured by the following grammar:

$$G_{c_n}: \quad \begin{array}{l} S \rightarrow bS\overline{b} \mid A \\ A \rightarrow AB \mid \epsilon \\ B \rightarrow bB\overline{b} \mid \epsilon \end{array} \tag{9}$$

In fact, the three structures of (8) are all plausible ones as regards RNA folding, and moreover cases of *alternative secondary structure* observed *in vivo* can be seen to be well-modelled by the ambiguous grammar G_o. The deterministic grammar G_{od}, however, captures only one secondary structure for any given string – in this case, the dumbbell structure. Thus, it does not adequately represent the inherent ambiguity of strings of the form $L_a = \{u\overline{u}^R u\overline{u}^R \mid u \in \Sigma_{\mathrm{DNA}}^*\}$, which is a subset of both L_h and L_d (and in fact is the intersection of either with the copy language). We call L_a the *attenuator* language, because it models a bacterial regulatory system that employs alternative secondary structures to form a binary switch *in vivo* [18]. The simpler but non-ideal attenuator language $L_{an} = \{u\overline{u}^R u \mid u \in \Sigma_{\mathrm{DNA}}^*\}$ is another version of this phenomenon, where either of the u's can and does base-pair with the central $\overline{u}^R$ in different circumstances, demonstrating both the importance of ambiguity in this domain and the natural correspondence between the formal linguistic notion and the biological theme of alternative secondary structure.

While the grammar G_o can thus be seen to more adequately capture desirable structural features, there is also a sense in which it is overly ambiguous with respect to the biological domain. This can be seen in the last two derivations in (8), which differ only in which of the first pair of doubled S's is chosen to be doubled again. Either case leads to the same cruciform structure, thus we call this grammar *structurally ambiguous*. As an alternative, we propose the following grammar which generates each possible secondary structure for a given string exactly once:

$$G_{on}: \quad \begin{array}{l} S \rightarrow A \mid \varepsilon \\ A \rightarrow bA\overline{b} \mid AB \mid b\overline{b} \\ B \rightarrow bA\overline{b} \mid b\overline{b} \end{array} \tag{10}$$

These results are summarized in the following lemma, where we consider "secondary structure" to be the pairings of bases that appear together in a single step of the derivation:

Lemma 2.2 (Structural Ambiguity) *For the grammars and language of orthodox strings defined above, $L(G_o) = L(G_{od}) = L(G_{on}) = L_o$. For any $w \in L_o$, there is exactly one leftmost derivation from G_{od}. For any secondary structure of any $w \in L_o$, there is exactly one leftmost derivation from G_{on}.*
Proof: By various inductions (e.g. see [18]). □

Perhaps the most significant result to emerge from this line of research has been that DNA is beyond context-free. This follows from several observations:

- Direct repeats are common in DNA, and copy languages, of which $L_a = \{uu \mid u \in \Sigma^*\}$ is the archetype, are known not to be context-free. Of course, the mere presence of repeats is not a proof (a fact often overlooked), since in fact repeats are required in context-free languages by the pumping lemma. Thus, it is necessary to establish functional roles for specific repeats in order to make a formal argument; a number of possibilities have been proposed [18]. For example, the attenuator languages L_a and L_{an} introduced above are non-context-free by virtue of repeats with one such functional role, founded in the need for alternative secondary structure in certain control mechanisms.

- More generally, crossing dependencies are observed in parallel (as opposed to anti-parallel) interactions between strands, seen commonly in proteins and less commonly in phenomena such as triple- and quadruple-stranded DNA.

- *Pseudoknots* are a form of secondary structure in which two stem-and-loop structures overlap, such that one of the loops contains half of the other stem. The ideal version of the pseudoknot language, $L_\psi = \{uv\overline{u}^R\overline{v}^R \mid u, v \in \Sigma^*_{\mathrm{DNA}}\}$, contains no repeats but, while each of the stems is conventionally based-paired with nested dependencies, in combination the dependencies are forced to cross. Pseudoknots, and non-orthodox secondary structure in general, have created challenges for algorithms dealing with secondary structure prediction and pattern recognition.

Thus the language of DNA appears to be relatively complex in a formal linguistic sense, and the question arises as to how such complexity arises. That is, we might presume that the first DNA (or, more likely, RNA) molecules were random strings, and thus regular, and ask by what series of operations such strings were manipulated to create the complex languages which have evidently been selected by evolution. The mathematical way of asking this rather philosophical question might be in terms of closure properties under the various domain-specific operations that are observed on strings of DNA. Along these lines, we have observed the following:

- Under the operation of *replication*,

  $$\mathrm{REP}(L) = \{w, \overline{w}^R \mid w \in L\} = L \cup \overline{L}^R \tag{11}$$

  all the classes of the Chomsky hierarchy are closed (since they are closed under the individual operations of homomorphism, reversal, and union). In fact, we observe a fixpoint:

  $$\mathrm{REP}^*(L) = \mathrm{REP}(L) \tag{12}$$

  However, it is noteworthy that deterministic context-free languages, e.g.

  $$L_{det} = \{\mathrm{g}^i \mathrm{a}^j \mathrm{t}^k \mathrm{c}^* \mid i = j + k\} \tag{13}$$

  are not closed under replication, since we observe that

  $$\mathrm{REP}(L_{det}) = \{\mathrm{g}^p \mathrm{a}^q \mathrm{t}^r \mathrm{c}^s \mid p = q + r \;\text{ or }\; s = q + r\} \tag{14}$$

  is not only nondeterministic but inherently ambiguous, necessarily having multiple leftmost derivations whenever $p = q + r = s$.

- The classifications of the Chomsky hierarchy are also closed under recombinational operations such as ligation (and its closure)

  $$\mathrm{LIG}(L) = \{xy \mid x, y \in L\} = L \cdot L \tag{15}$$

  since DNA can only ligate head-to-tail, as well as under scission (and its closure)

  $$\begin{aligned} \mathrm{CUT}(L) &= \{x, y \mid xy \in L\} = \mathrm{PRE}(L) \cup \mathrm{SUF}(L) \\ \mathrm{CUT}^*(L) &= \{u \mid xuy \in L\} = \mathrm{PRE}(\mathrm{SUF}(L)) \end{aligned} \tag{16}$$

since they are closed under the operations of prefix and suffix.[3] Even though context-free languages as a whole are closed under scission, once again this is not the case for deterministic context-free languages, nor for unambiguous languages. Although we cannot directly model ligation which circularizes strings, we can model their scission:

$$\mathrm{CUT}(\mathrm{LIG}^{\bigcirc}(L)) = \{vu \mid uv \in L\} = \mathrm{CYC}(L) \tag{17}$$

Context-free languages are known to be closed under this operation of cyclic permutations [10].

- This picture changes, however, when we examine evolutionary rearrangements, that involve block movements of substrings:

$$\begin{aligned} \mathrm{DUP}(L) &= \{xuuy \mid xuy \in L\} \\ \mathrm{INV}(L) &= \{x\overline{u}^R y \mid xuy \in L\} \\ \mathrm{XPOS}(L) &= \{xvuy \mid xuvy \in L\} \\ \mathrm{DEL}(L) &= \{xy \mid xuy \in L\} \end{aligned} \tag{18}$$

  where $x, y, u, v \in \Sigma^*_{\mathrm{DNA}}$ and $L \subseteq \Sigma^*_{\mathrm{DNA}}$. Regular or context-free languages could not be closed under duplication, since this creates copy languages; neither are they closed under inversion (which makes copy languages from inverted repeats) or transposition (which makes pseudoknots from them). (See [18] for formal proofs). Only under deletion are the lower levels of the Chomsky hierarchy preserved – thus, evolution may tend towards increasing linguistic complexity, in this purely formal sense.

- Another source of formal linguistic complexity may arise from the fact that context-free languages are not closed under intersection; for example, the pseudoknot language L_ψ can be seen to be the intersection of two context-free stem-and-loop languages. During gene expression, transcription, processing, and translation may take place at different times and/or in different compartments of the cell. Thus, the signals

[3]Scission and ligation play an important role in *splicing systems,* language-theoretic constructs which were inspired by restriction enzymes and DNA. These have been studied extensively by Tom Head [9] and others [3, 5, 6, 11, 12, 25, 26].

relevant to the DNA, various forms of RNA, and protein, are all projected back to the DNA, and to the extent these can or should be viewed as separate languages, the DNA must be seen as the *intersection* of those languages. In addition, there is evidence that secondary structure may play a role in expression (e.g. regulating alternative splicing), and in fact it may interfere with ribosome binding – and context free languages are also not closed under complementation.

3 Intermolecular Structure

That macromolecules interact with each other is fundamental to the workings of biological systems. Thus, the traditional formal view of languages can be seen to be seriously handicapped, insofar as it captures dependencies within strings but not *between* strings in a collection. The author has recently examined extensions that might make it possible to formalize such intermolecular interactions [22].

A *cut grammar* is an ordinary grammar which in addition has a new symbol δ not otherwise in the grammar, which may appear on the right-hand sides of productions. Derivations are as for ordinary grammars, producing sentential forms with (possibly) interspersed δ's. As always, we have the ordinary language $L(G) = \{\, w \in (\Sigma \cup \{\delta\})^* \mid S \Longrightarrow^* w\}$, but in general derivations from cut grammars require a final modification:

Definition 3.1 (Cut Languages) *For any* $w = w_1\delta w_2\delta\cdots\delta w_n$ *where* $w_i \in \Sigma^*$ *for* $1 \leq i \leq n$, *we define a* cut function $\widehat{w} = \{w_1, w_2, \cdots, w_n\}$ *and an* uncut function $\widetilde{w} = w_1 w_2 \cdots w_n$. *For a cut grammar* G *with start symbol* S *we define the* cut language $\widehat{L}(G) = \{\, \widehat{w} \in 2^{\Sigma^*} \mid S \Longrightarrow^* w\}$ *and the* uncut language $\widetilde{L}(G) = \{\, \widetilde{w} \in \Sigma^* \mid S \Longrightarrow^* w\}$.

The cut language union, $\bigcup \widehat{L}(G) = \{\, u \in \Sigma^* \mid S \Longrightarrow^* w \text{ and } u \in \widehat{w}\}$, will also be important. A string will be in this language if and only if it is a substring of some string derived from the grammar in the ordinary way, such that the substring is bracketed by δ's and/or the termini of the string. For reasons that will become apparent, we will be more interested in the cut languages of such grammars, which are sets of sets of strings related by a particular derivation. For a cut grammar in which no δ appears we have

$L(G) = \widetilde{L}(G) = \bigcup \widehat{L}(G)$. For the double-strand grammar $G_{ds} : S \to bS\overline{b} \mid \delta$, we see that

$$\widehat{L}(G_{ds}) = \{ \{u, \overline{u}^R\} \mid u \in \Sigma^*_{\mathrm{DNA}} \} \tag{19}$$

which is related to the hairpin language by

$$L_h = \widetilde{L}(G_{ds}) = \{ u \mid \{u\} \in \widehat{L}(G_{ds})\}, \tag{20}$$

the latter equality being a consequence of Lemma 1.3.

It is not immediately obvious that adding cuts does not increase the linguistic complexity of the various sets we will deal with, but this is in fact the case:

Theorem 3.1 (Cut Language Closure) *For any cut grammar G, if the ordinary language $L(G)$ is regular (context-free, recursively enumerable), then so are $\widetilde{L}(G)$, each set in $\widehat{L}(G)$, and $\bigcup \widehat{L}(G)$.*

Proof: ($\widetilde{L}(G)$): For any G we can change each δ to ϵ, calling this ordinary grammar $\widetilde{G}$; clearly $L(\widetilde{G}) = \widetilde{L}(G)$.

($\widehat{L}(G)$): This follows vacuously, since every set in $\widehat{L}(G)$ contains substrings of a finite string and so must itself be finite, therefore regular, regardless of the nature of $L(G)$.

($\bigcup \widehat{L}(G)$): We have previously demonstrated this by closure under the operation of finite transducers [22]. Since then Tilman Becker has developed an elegant constructive proof for the case of a context-free cut grammar G [unpublished]. This involves creating an ordinary context-free $\widehat{G}$ that effectively chooses each pair of adjacent δ's in any derivation and generates only the intervening substring, so that $\cup \widehat{L}(G) = L(\widehat{G})$. $\square$

While the ability to model double-stranded DNA is a modest improvement, we can elaborate on this to deal with increasingly complex molecular assemblages. *Nicked* double-stranded DNA can be modelled by

$$S \to bS\overline{b} \mid \delta S \mid S\delta \mid \delta \tag{21}$$

We can also require a minimum "overhang" to create what biologists call "sticky ends":

$$S \to bS\overline{b} \mid w\delta S\overline{w}^R \mid wS\delta\overline{w}^R \mid \delta \tag{22}$$

for each $w \in \Sigma^n_{\mathrm{DNA}}$ where n is the desired length (or for particular w's to model restriction enzyme sites). Thus, cut grammars can be used to describe

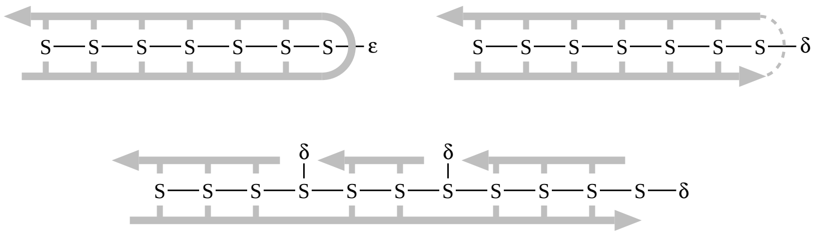

Figure 3: A hairpin, double-stranded DNA, and nicked DNA.

hybridization of populations of strings, e.g. cut language elements as sets of hybridizable oligonucleotides. Note that this formalizes the strategy recently used by Adleman to "compute" a well-known intractable problem [1].

This is easily generalized to branched hybridization, again by analogy with the development in the previous section. The language of generalized hybridization networks would derive from

$$S \to bS\overline{b} \mid SS \mid \delta \mid \epsilon \tag{23}$$

since nicks can thus arise at the end of any stem or, via the doubled SS rule, in-line on either side of a stem.

Still other generalizations suggest themselves. Note, for example, that using the start symbol to leave one end of the double-stranded molecule open, and a δ to cut open the other end, seems arbitrary given the symmetry. In order to "close off" the start of a derivation tree, we can do the following:

Definition 3.2 (Circular Cut Languages) *For any* $u = u_1\delta u_2\delta u_3\delta\cdots\delta u_n$ *where* $n > 1$ *(i.e. at least one* δ *appears in* u*) and* $u_i \in \Sigma^*$ *for* $1 \leq i \leq n$*, we define a* circular cut function $\mathring{u} = \{u_n u_1, u_2, u_3, \cdots, u_{n-1}\}$. *A* circular cut language *is defined as before, but using the circular cut function.*

Then, for $G_o : S \to bS\overline{b} \mid \delta$, we have ordinary stems

$$\bigcup \mathring{L}(G_o) = \widetilde{L}(G_o) = L_o \tag{24}$$

which is to say, the set of stems open at the start is the same as the set open at the terminus. For any G we can form a G' by adding a new start symbol S' and rule $S' \to S\delta$, to get $\mathring{L}(G') = \widehat{L}(G)$.

More generally, circular cut languages will allow us to begin derivations at interior points in structures. The following grammar forms a hybridization "wheel" with an arbitrary number of spokes radiating from a central S:

$$\begin{aligned} S &\rightarrow\ AS \mid \epsilon \\ A &\rightarrow\ bA\overline{b} \mid \delta \end{aligned} \tag{25}$$

In order to be able to distinguish between "ligateable" cuts (which is to say, nicks) and unligateable gaps, we introduce the following:

Definition 3.3 (Ligation) *A* ligation grammar *is a cut grammar with an additional new symbol γ. For any $u = u_1\gamma u_2\gamma\cdots\gamma u_n$ where $u_i \in (\Sigma \cup \{\delta\})^*$ for $1 \le i \le n$, we define a* ligate function $\breve{u} = \{\widetilde{u}_1, \widetilde{u}_2, \cdots, \widetilde{u}_n\}$. *For a ligation grammar G with start symbol S we define the* ligated language $\breve{L}(G) = \{\ \breve{u} \in 2^{\Sigma^*} \mid S \Longrightarrow^* u\}$.

This allows us to propose a model of *generalized linear hybridization*, as embodied in the grammar below:

$$G_{lh}: \quad \begin{aligned} S &\rightarrow\ bS\overline{b} \mid \delta A \mid B\delta \mid \gamma \\ A &\rightarrow\ \gamma Ab \mid bS\overline{b} \mid \gamma \\ B &\rightarrow\ bB\gamma \mid bS\overline{b} \mid \gamma \end{aligned} \tag{26}$$

It can be seen that any cut appearing at the end of a stem is an unligateable gap (γ), as are cuts opposite unpaired bases, in the A and B rules. Cuts appearing between paired bases, via the S rule, are ligateable nicks (δ). Such a grammar can produce parse trees representing plausible linear hybridization products such as that shown in Figure 4.

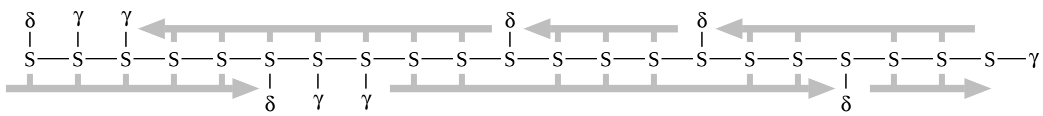

Figure 4: A linear hybridization parse tree.

Given such a model, the problem of determining ways in which a set of oligonucleotides can anneal can be cast as a parsing task, albeit a non-traditional one. Unfortunately, even for the case of "ideal" sets (in the sense that every base can potentially pair, though between rather than within strings), this task appears to be intractable. The following proof is due to Michael Niv:

Lemma 3.1 (Cut Language Recognition) *Given a cut grammar G and a set $V \subseteq \Sigma^*$, determining whether $V \in \widehat{L}(G)$ is NP-hard.*
Proof: By reduction from Directed Hamiltonian Path (DHP). Given a graph (V, E), where $|V| = n$, the start vertex is v_1, and the end vertex is v_n, if we define a cut grammar with the vertex labels as its alphabet ($\Sigma = V$), nonterminals $\{A^i_j \mid 1 \leq i, j \leq n\} \cup \{S\}$, and rules

$$\begin{aligned} S &\rightarrow v_1 A^1_1 \\ A^i_j &\rightarrow \delta v_k A^{i+1}_k \quad \text{for all } (j,k) \in E,\ 1 \leq i < n \\ A^n_n &\rightarrow \epsilon \end{aligned} \tag{27}$$

then $V \in \widehat{L}(G)$ *iff* there is a path of length n that passes through every vertex in V exactly once (DHP). □

4 Evolutionary Structure

Just as conventional grammars fall short in describing interactions between strings in a set, they are also inadequate to capture evolutionary relationships between strings. Formal language theory has been notably absent in one of the most widely-practiced computational activities surrounding macromolecules, that of string comparison. The determination of optimal alignments and putative evolutionary distances between strings has heretofore been confined to the realm of algorithmics.

The close relationship between grammars and automata is an important aspect of formal language theory. We have recently explored the use of a brand of automaton called a finite transducer in modelling relationships between strings in such a way as to provide a connection to the efficient algorithms commonly used in this field [24].

Finite transducers are simply finite-state machines for which transitions have both input and output. Figure 5a shows how a finite transducer can be used to model a single mutation occurring anywhere in a string, where the mutation could be a single-base substitution, deletion, or insertion (using

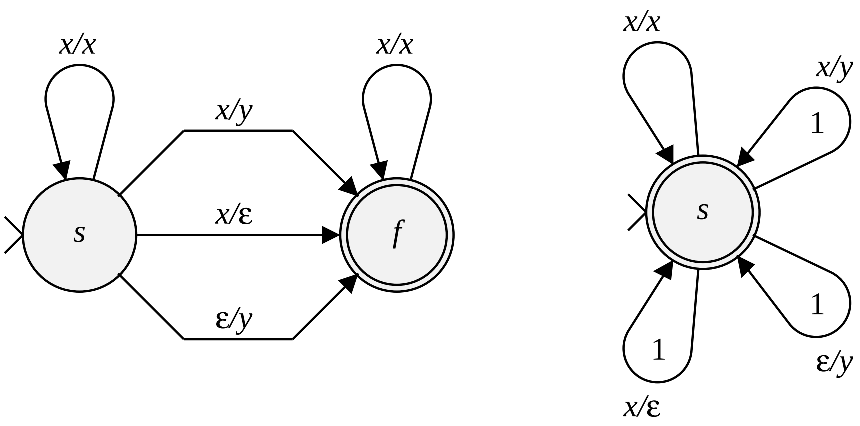

Figure 5: Finite transducers for single mutation (left) and edit (right).

transitions labelled by their input and output, separated by a foreslash, with x and y standing for any non-identical nucleotides).

By simply merging the start and final states of the "mutation machine", we can produce an "edit machine" as shown in Figure 5b, that will make any number of non-overlapping mutations in a string. We have also introduced *weights* on the transitions, to be added to a running total with each move of the transducer. What is conventionally defined as the minimal edit distance between two strings is simply the minimal computation of this automaton. This affords the following formulation of the notion of edit distance:

Definition 4.1 (Edit Distance) *Consider a weighted finite transducer* $\langle Q, \Sigma, \delta, s, F\rangle$ *with states* Q*, input/output alphabet* Σ*, start state* $s \in Q$*, final states* $F \subseteq Q$*, and transitions of the form* $\delta \subset Q \times \Sigma^* \times \Sigma^* \times \Re \times Q$*, where* $\Re$ *is the set of reals. Given a pair of strings* $x, y \in \Sigma^*$*, for each state* $q_i \in Q$ *construct a matrix* $q_i[0..|x|, 0..|y|]$ *of reals such that*

- *if* $q_i \in F$*,* $q_i[0,0] = 0$
- *for* $0 \leq a \leq |x|$ *and* $0 \leq b \leq |y|$*,*

$$q_i[a,b] = \min \{\ q_j[a-|u|, b-|v|\,] + n \mid \langle q_i, u, v, n, q_j\rangle \in \delta,\ u = x_{(a-|u|+1)..a} \text{ and } v = y_{(b-|v|+1)..b}\}$$

Then the edit distance *between* x *and* y *is* $s[\,|x|, |y|\,]$.

That is to say, we consider each state of the transducer to be a matrix, and each term in the recurrence defining that matrix to be determined by outgoing transitions from that state. In fact it may be said that the automaton is simply an alternative form of the mathematical recurrence. Applying Definition 4.1 to the edit transducer produces the classic edit distance recurrence:

$$s[i,j] = \min \begin{cases} s[i-1,j-1] & \text{if } x_i = y_j \\ s[i-1,j-1]+1 & \text{if } x_i \neq y_j \\ s[i-1,j]+1 & \\ s[i,j-1]+1 & s[0,0]=0 \end{cases} \tag{28}$$

It has long been noted that simple edit distance is not a realistic model of biological mutation, insofar as insertions and deletions tend to involve more than one base at a time, and thus should not be penalized strictly in proportion to length. So-called *affine* gaps are modelled as in Figure 6, for gap initiation penalty α and gap extension penalty β.

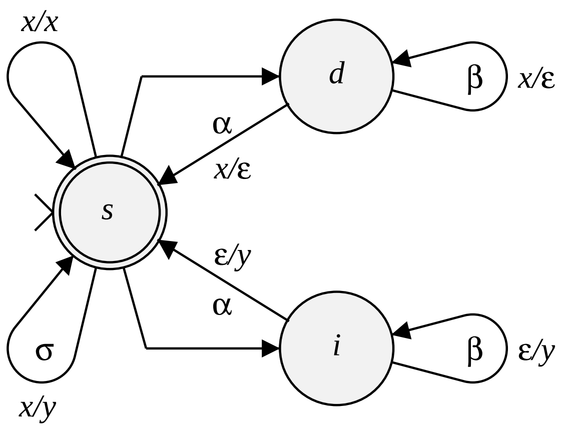

Figure 6: An affine gap transducer.

Again, applying Definition 4.1 to the transducer yields a well-known recurrence, first derived algebraically (and rather less concisely) by Gotoh [7]:

$$s[a,b] = \min \begin{cases} s[a-1,b-1] & \text{if } \; x_a = y_b \\ s[a-1,b-1]+\sigma & \text{if } \; x_a \neq y_b \\ d[a,b] & \\ i[a,b] & \qquad s[0,0]=0 \end{cases} \tag{29}$$

$$d[a,b] = \min \begin{cases} s[a-1,b]+\alpha \\ d[a-1,b]+\beta \end{cases} \qquad i[a,b] = \min \begin{cases} s[a,b-1]+\alpha \\ i[a,b-1]+\beta \end{cases}$$

More practical algorithms employ maximum similarity rather than minimum distance, and attempt to find local regions of similarity rather than global matches. Such algorithms can be modelled first by simply changing *min* to *max* in Definition 4.1 and reassigning weights accordingly, and second by adding unweighted *scanning* transitions to find selected local regions. A transducer for local alignment is illustrated in Figure 7. Here, only the transitions from the e state will be of relevance to the alignment.

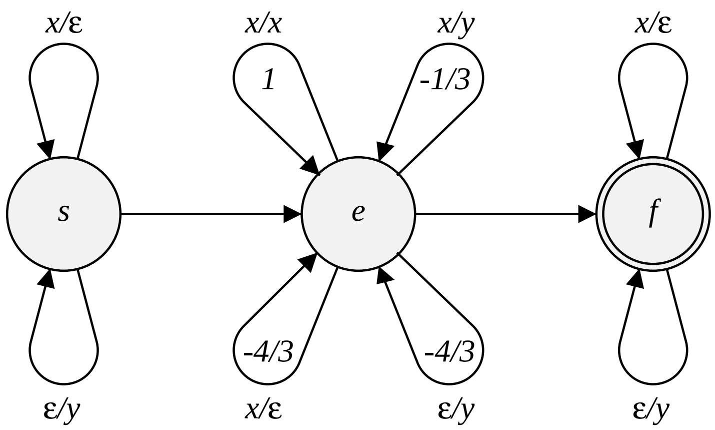

Figure 7: A local alignment transducer.

We can again apply a modified form of Definition 4.1 to the transducer, this time taking advantage of some simplifications, that follow by reasoning from the automaton:

$$s[a,b] = \max \begin{cases} s[a-1,b] \\ s[a,b-1] \\ e[a,b] \end{cases} = \max_{\substack{0 \le i \le a \\ 0 \le j \le b}} e[i,j]$$

$$e[a,b] = \max \begin{cases} e[a-1,b-1] + 1 & \text{if } x_a = y_b \\ e[a-1,b-1] - 1/3 & \text{if } x_a \neq y_b \\ e[a-1,b] - 4/3 \\ e[a,b-1] - 4/3 \\ f[a,b]\ (=0) \end{cases} \tag{30}$$

$$f[a,b] = \max \begin{cases} f[a-1,b] \\ f[a,b-1] \end{cases} = 0$$

We know that $f[a,b] = 0$ for any a, b because all out-transitions from f have zero weight, and any inputs can be emptied to achieve the conditions for termination. Similarly, s permits any prefixes of the inputs to be consumed with zero weight, so that the maximum weight from any position on the inputs is simply the maximum of zero and the result of the free transition from there to e (i.e. the maximum value in the matrix of e) which we have seen is at least zero. These are the same equations derived by Smith and Waterman [27]. Other uses of meta-alignment scanning nodes include best-fit alignments that specify containment or overlap as required by fragment assembly algorithms.

Such transducers can be transformed into *two-tape* transducers, where the input and output instead become a pair of inputs; then the output can be used to record the nature of individual moves, for example displaying individual columns of the resulting alignment. Such machines are termed *alignment* machines, and are especially useful in isolating local alignments as in the previous example, by producing empty output on scanning transitions.

Using automata to specify algorithms in this way invites much greater flexibility in design, and we have proposed a number of new formulations [24]. Among them are the aligner of Figure 8, which maintains a notion of correct reading frame. By penalizing mismatches *or* matches in one of three frames less than in others, it can effectively find the correct reading frame, when allowed to start form any state. The recurrence is as follows:

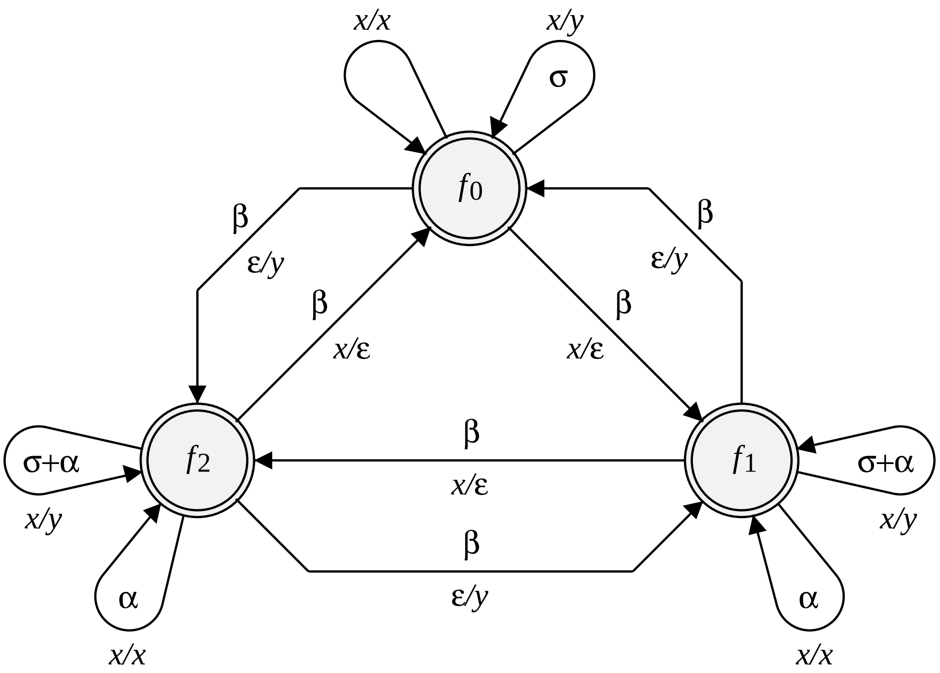

Figure 8: A frame-maintainence transducer.

$$f_k[i,j] = \min \begin{cases} f_k[i-1,j-1]+\alpha_k & \text{if } x_i = y_j \\ f_k[i-1,j-1]+\alpha_k+\sigma & \text{if } x_i \neq y_j \\ f_{(k+1) \bmod 3}[i-1,j]+\beta & \\ f_{(k+2) \bmod 3}[i,j-1]+\beta & \end{cases} \tag{31}$$

for $k = 0, 1, 2$, where 0 is the correct frame, i.e. $\alpha_0 = 0$ and $\alpha_1 = \alpha_2 = \alpha > 0$.

We have taken advantage of the relative ease with which novel alignment algorithms can be designed, by creating a visual programming system that makes use of a domain-specific drawing tool to specify an aligner, which is then translated automatically to code for the corresponding dynamic programming algorithm [24]. It is hoped that this increased ease of design and experimentation will encourage the development of new algorithms that incorporate specific domain knowledge, in what we call *model-based alignment*. Recently we have created an alignment machine that entails a model of gene structure, coded as an automaton rather than a grammar. With it we have been successful at comparing tubulin genes with disparate intron/exon structures, in such a way that gene structure is predicted with greater accuracy by means of the mutual information between two related genes.

5 Conclusion

In this review we have concentrated on linguistic aspects of biological macromolecules that are of some mathematical interest. In doing so, at least two major aspects of this area have been neglected. First, we have previously described grammars for the specification and recognition of the *informational* structure of DNA [16, 18, 20, 23]. This has involved the use of grammars and parsers in *syntactic pattern recognition*, and has led to actual implementations that are of use in recognizing genes [4] and other higher-order features [19]. By contrast, the grammars described in this paper deal with structural dependencies that arise purely by virtue of the biochemistry of nucleic acids, and not any information they encode.

The second area has been neglected because very little has been done to date, yet it promises to be a much richer domain of inquiry. This is the use of linguistic principles to describe proteins. Proteins not only have a larger alphabet than nucleic acids, but the range of possible interactions between residues is far richer than the simple base-pairing with which we have dealt. Departures from both ideality and orthodoxy are the norm, so that proteins will be much more challenging entities for modelling via grammars. However, an effort in this regard, extending the simple model of nucleic acids, could yield far more practical results. Representing the interactions in proteins that arise by way of folding is a step toward a compositional semantics that might go some ways toward encompassing function. The lucidity and firm formal foundations of linguistic methods could have great benefit in this very complex domain.

6 Acknowledgements

This work was supported by the US Department of Energy under grant number 92ER61371, and by the National Institutes of Health under grant number P50HG00425. The author gratefully acknowledges the contributions of G. Christian Overton, Dominic Bevilacqua, Michael Niv, Tilman Becker, Aravind Joshi, Kevin Murphy, Sandip Biswas, and Jim Tisdall.

References

[1] L.M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 1994.

[2] E. Chargaff. Chemical specificity of nucleic acids and mechanism for their enzymatic degradation. *Experientia*, 6:201–209, 1950.

[3] K.L. Denninghoff and R.W. Gatterdam. On the undecidability of splicing systems. *International Journal of Computer Mathematics*, 27:133–145, 1989.

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

[5] R.W. Gatterdam. Splicing systems and regularity. *International Journal of Computer Mathematics*, 31:63–67, 1989.

[6] R.W. Gatterdam. Algorithms for splicing systems. *SIAM Journal of Computing*, 21:507–520, 1992.

[7] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.

[8] L. Grate, M. Herbster, R. Hughey, I.S. Mian, H. Noller, and D. Haussler. RNA modeling using Gibbs sampling and stochastic context free grammars. In *Proc. of Second Int. Conf. on Intelligent Systems for Molecular Biology*, Menlo Park, CA, August 1994. AAAI/MIT Press.

[9] T. Head. Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. *Bull. Math. Biol.*, 49(6):737–759, 1987.

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

[11] K. Culik II and T. Harju. The regularity of splicing systems and DNA. In *16th International Colloquium on Automata Languages and Programming (Lecture Notes in Computer Science 372)*, pages 222–233. Springer-Verlag, Berlin, 1989.

[12] K. Culik II and T. Harju. Splicing semigroups of dominoes and DNA. *Discrete Applied Mathematics*, 31:261–277, 1991.

[13] J. Josse, A.D. Kaiser, and A. Kornberg. Enzymatic synthesis of deoxyribonucleic acid. VIII. Frequencies of nearest neighbor base sequences in deoxyribonucleic acid. *J. Biol. Chem.*, 236:864–875, 1961.

[14] Y. Sakakibara, M. Brown, R. Hughey, I.S. Mian, K. Sjölander, R.C. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Res.*, 22(23):5112–5120, 1994.

[15] Y. Sakakibara, M. Brown, R. Hughey, I.S. Mian, K. Sjölander, R.C. Underwood, and D. Haussler. Recent methods for RNA modeling using stochastic context-free grammars. In *Proceedings of the Asilomar Conference on Combinatorial Pattern Matching*, New York, NY, 1994. Springer-Verlag.

[16] D. B. Searls. Representing genetic information with formal grammars. In *Proceedings of the National Conference on Artificial Intelligence*, pages 386–391. American Association for Artificial Intelligence, 1988.

[17] D. B. Searls. Investigating the linguistics of DNA with definite clause grammars. In E. Lusk and R. Overbeek, editors, *Logic Programming: Proceedings of the North American Conference*, pages 189–208. MIT Press, 1989.

[18] D. B. Searls. The computational linguistics of biological sequences. In L. Hunter, editor, *Artificial Intelligence and Molecular Biology*, chapter 2, pages 47–120. AAAI Press, 1993.

[19] D. B. Searls and S. Dong. A syntactic pattern recognition system for DNA sequences. In H. A. Lim, J. Fickett, C. R. Cantor, and R. J. Robbins, editors, *Proceedings of the 2nd International Conference on Bioinformatics, Supercomputing, and Complex Genome Analysis*, pages 89–101. World Scientific, 1993.

[20] D. B. Searls and M. O. Noordewier. Pattern-matching search of DNA sequences using logic grammars. In *Proceedings of the Conference on Artificial Intelligence Applications*, pages 3–9. IEEE, 1991.

[21] D.B. Searls. The linguistics of DNA. *American Scientist*, 80(6):579–591, 1992.

[22] D.B. Searls. Formal grammars for intermolecular structure. *First International IEEE Symposium on Intelligence in Neural and Biological Systems*, pages 30–37, 1995.

[23] D.B. Searls. String variable grammar: A logic grammar formalism for the biological language of DNA. *Journal of Logic Programming*, 24(1,2):73–102, 1995.

[24] D.B. Searls and K. Murphy. Automata-theoretic models of mutation and alignment. *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, 1995.

[25] R. Siromoney, K.G. Subramanian, and V.R. Dare. Circular DNA and splicing systems. In *Parallel Image Analysis (Lecture Notes in Computer Science 654)*, pages 260–273. Springer-Verlag, Berlin, 1992.

[26] R. Siromoney, K.G. Subramanian, and V.R. Dare. On identifying DNA splicing systems from examples. In *(Lecture Notes in Artificial Intelligence 642)*, pages 305–319. Springer-Verlag, Berlin, 1992.

[27] T.F. Smith and M.S. Waterman. Identification of common molecular sequences. *J. Mol. Biol.*, 147:195–197, 1981.

[28] J.D. Watson and F.H.C. Crick. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, 171:737–738, 1953.