Molecules, Languages and Automata

David B. Searls

Lower Gwynedd, PA 19454, USA

Abstract. Molecular biology is full of linguistic metaphors, from the language of DNA to the genome as "book of life." Certainly the organization of genes and other functional modules along the DNA sequence invites a syntactic view, which can be seen in certain tools used in bioinformatics such as hidden Markov models. It has also been shown that folding of RNA structures is neatly expressed by grammars that require expressive power beyond context-free, an approach that has even been extended to the much more complex structures of proteins. Processive enzymes and other "molecular machines" can also be cast in terms of automata. This paper briefly reviews linguistic approaches to molecular biology, and provides perspectives on potential future applications of grammars and automata in this field.

1 Introduction

The terminology of molecular biology from a very early point adopted linguistic and cryptologic tropes, but it was not until some two decades ago that serious attempts were made to apply formal language theory in this field. These included efforts to model both the syntactic structure of genes, reflecting their hierarchical organization, and the physical structure of nucleic acids such as DNA and RNA, where grammars proved suitable for representing folding patterns in an abstract manner. In the meantime, it was also recognized that automata theory could be a basis for representing some of the key string algorithms used in the analysis of macromolecular sequences. These varied approaches to molecular biology are all bound together by formal language theory, and its close relationship to automata theory. In reviewing these approaches, and discussing how they may be extended in new directions within biology, we hope to demonstrate the power of grammars as a uniform, computer-readable, executable specification language for biological knowledge.

2 Structural Grammars

Nucleic acids are polymers of four bases, and are thus naturally modeled as languages over the corresponding alphabets, which for DNA comprise the well-known set $\Sigma = \{\mathrm{a}, \mathrm{c}, \mathrm{g}, \mathrm{t}\}$. (RNA bases are slightly different, but for all practical purposes can be treated the same.) DNA, which carries the genetic information in our chromosomes, tends to form itself into a double helix with two strands

J.M. Sempere and P. García (Eds.): ICGI 2010, LNAI 6339, pp. 5–10, 2010.

that are held together by a complementary pairing of the opposing bases, 'a' with 't' and 'g' with 'c'. RNA molecules are more often single-stranded, though they can fold back on themselves to form regions of double-stranded structure, called *secondary structure*.

Given these bare facts, the language of all possible RNA molecules is specified by the following trivial grammar (with S the start symbol and ϵ the empty string, as usual):

$$S \rightarrow xS \mid \epsilon \qquad \text{for each } x \in \{\mathrm{a}, \mathrm{c}, \mathrm{g}, \mathrm{u}\} \tag{1}$$

It would seem to be natural to specify DNA helices, which comprise two strands, as a pair of strings, and a DNA language as a set of such pairs. However, DNA has an additional important constraint, in fact two: first, the bases opposing one another are complementary, and second, the strings have directionality (which is chemically recognizable) with the base-paired strands running in opposite directions.

We have previously shown how this sort of model can be extended to describe a number of specific phenomena in RNA secondary structure, and in fact a simple addition to the stem-and-loop grammar allows for arbitrarily branching secondary structures:

$$S \rightarrow\ xS\bar{x} \mid SS \mid \epsilon \qquad \text{where } \bar{g}{=}c,\ \bar{c}{=}g,\ \bar{a}{=}t,\ \bar{t}{=}a \tag{2}$$

Examples of such branching secondary structure include a cloverleaf form such as is found in *transfer RNA* or *tRNA*, an important adaptor molecule in the translation of genetic information from messenger RNA to protein. The language of (2) describes what is called *orthodox secondary structure*, which for our purposes can be considered to be all fully base-paired structures describable by context-free grammars.

There are, however, secondary structures that are beyond context-free, the archetype of which are the so-called *pseudoknots*. Pseudoknots can be conceived as a pair of stem-loop structures, one of whose loops constitutes one side of the others stem. The corresponding (idealized) language is of the form $uv\bar{u}^R\bar{v}^R$, which cannot be expressed by any context-free grammar. It is sometimes described as the intersection of two context-free palindromes of the form $uv\bar{u}^R$ and $v\bar{u}^R\bar{v}^R$, but of course context-free languages are not closed under intersection.

Pseudoknots and other non-context-free elements of the language of secondary structure can be easily captured with context-sensitive grammars, but the resulting complex movements of nonterminals in sentential forms tend not to enlighten. Rather, grammars with more structured rules, such as Tree-Adjoining Grammars (TAG), have been more profitably used for this purpose [6].

A grammar variation that the author has proposed describes even more complex, multi-molecular base-paired complexes of nucleic acids [4]. This is enabled by the addition to any grammar of a new symbol δ which is understood to cut the string at the point it appears. This means that derivations ultimately give rise not to strings but to sets of strings arising from such cuts, which may be base-paired in arbitrarily ramified networks.

Proteins are more complex macromolecular structures with several kinds of intermolecular interactions. Some of the basic types of such recurrent structural themes have been described with a variety of grammars [6].

3 Gene Grammars

Genes, which are encoded in the DNA of organisms, have a hierarchical organization to them that is determined by the process by which they are converted into proteins (for the most part). Genes are first *transcribed* into messenger RNA, or mRNA, which constitutes a complementary copy of the gene, and then this is *translated* into protein. The latter step requires the DNA/RNA code to be adapted to that of proteins, whose alphabet comprises the twenty amino acids. This encoding is called the *genetic code*, which appears as a table of triplets of bases mapped to amino acids.

Transcription itself involves a number of complications regarding the structure of genes, such as the fact that the actual coding sequence is interrupted by segments that are spliced out at an intermediate step, establishing what is called the *intron/exon* structure of the gene. In addition there are many signal sequences embedded in the gene, including in flanking non-coding regions, that determine such things as the starting point of transcription, the conditions under which transcription will occur, and the points at which splicing will occur.

The author has demonstrated how grammars can effectively capture all these features of genes, including ambiguities such as alternative splicing whereby different versions of genes may arise from the same genome sequence [1]. Such grammars have been used to recognize the presence of genes in raw sequence data by means of parsing, in what amounts to an application of syntactic pattern recognition [2]. (Modern 'gene-finders', however, use highly customized algorithms for efficiency, though the most effective of these do capture the syntactic structure of the standard gene model.) As the variety of genes and related features (such as immunoglobulin superfamily genes and microRNA) and their higher-level organization in genomes continues to grow more complex, grammars may yet prove to be a superior means to formally specify knowledge about such structure.

4 Genetic Grammars

Gregor Mendel laid the foundation for modern genetics by asserting a model for the inheritance of traits based on a parsimonious set of postulates. While many modifications have been required to account for a wider and wider set of observations, the basic framework has proven robust. Many mathematical and computational formalizations of these postulates and their sequelae have been developed, which support such activities as pedigree analysis and genetic mapping.

The author has been developing a grammar-based specification of Mendelian genetics which is able to depict the basic processes of gamete formation, segregation of alleles, zygote formation, and phenotypic expression within a uniform

framework representing genetic crosses [unpublished]. With this basic 'Mendelian grammar,' extensions are possible that account in a natural way for various known mechanisms for modification of segregation ratios, linkage, crossing-over, interference, and so forth.

One possible use of this formalism is as a means to frame certain types of analysis as a form of grammar inference. For example, mapping of genes to linkage groups and ordering linkage groups can be seen as finding an optimal structure of an underlying grammar so as to fit experimental data. Especially intriguing is the possibility of including together in one framework the genetic analysis with phenotypic grammars, for example in the genetic dissection of pathways.

5 Molecular Machines

Enzymes and other biochemical structures such as ribosomes are sometimes called 'molecular machines' because they perform repetitive chemical and/or mechanical operations on other molecules. In particular, a large class of such objects process nucleic acids in various ways, many of them by attaching to and moving along the DNA or RNA in what is termed *processive* fashion. This immediately brings to mind computational automata which perform operations on tapes.

Since automata have their analogues in grammars, it is natural to ask whether grammars can model enzymes that act on DNA or RNA. In fact the trivial right-recursive grammar that we showed at the outset (1) can be considered a model for *terminal transferase*, an enzyme that synthesizes DNA by attaching bases to a growing chain, as in this derivation:

$$S \Rightarrow \mathrm{c}S \Rightarrow \mathrm{ct}S \Rightarrow \mathrm{ctc}S \Rightarrow \mathrm{ctca}S \Rightarrow \mathrm{ctcaa}S \Rightarrow \mathrm{ctcaag}S \Rightarrow \mathrm{ctcaag}$$

We can view the nonterminal S as the molecular machine, the terminal transferase itself, laying down bases sequentially and then departing into solution. Similarly, we can envision a context-sensitive grammar that models an *exonuclease*, an enzyme that degrades nucleic acids a base at a time from one or the other end. The orientation is important, because exonucleases are specific for which end they chew on, and therefore whether they run in the forward or reverse direction on the strand:

$$\begin{array}{ll} Fx \rightarrow F \mid \epsilon & \textit{forward exonuclease} \\ xR \rightarrow R \mid \epsilon & \textit{reverse exonuclease} \end{array} \tag{3}$$

These can produce derivations such as the following, with the nonterminals again physically mimicking the action of the corresponding enzymes:

$$F\mathrm{gcaa} \Rightarrow F\mathrm{gcaa} \Rightarrow F\mathrm{caa} \Rightarrow F\mathrm{aa} \Rightarrow F\mathrm{a} \Rightarrow F \Rightarrow \epsilon$$
$$\mathrm{atggac}R \Rightarrow \mathrm{atgga}R \Rightarrow \mathrm{atgg}R \Rightarrow \mathrm{atg}R \Rightarrow \mathrm{at}R \Rightarrow \mathrm{at}$$

In the first derivation, the F completely digests the nucleic acid strand and then itself disappears via the ϵ disjunct — into solution, as it were. On the other hand, in the second example we show the R exonuclease departing without completing the job, which mirrors the biological fact that enzymes can show greater or lesser propensity to hop on or off the nucleic acid spontaneously. We could model the tendency to continue the recursion (known as an enzyme's *processivity*) with a stochastic grammar, where probabilities attached to rules would establish the half-lives of actions of the biological processes.

The author's most recent efforts [unpublished] have been to catalogue a wide range of grammars describing the activities of enzymes acting on nucleic acids in various circumstances. This requires the extension of the model to double-stranded DNA, as well as the ability to act on more than one double-stranded molecule at once. Wth the employment of stochastic grammars, it appears possible to specify a wide variety of biochemical details of molecular machines.

6 Edit Grammars

Another view of the movement of nonterminals is as a means to perform editing operations on strings. As in the case for processive enzymes, we view a nonterminal as a 'machine' that begins at the left end of an input string and processes to the right end, leaving an altered string as output.

$$\begin{aligned} xS &\xrightarrow{0} Sx && \textit{identity } (x \in \Sigma) \\ yS &\xrightarrow{1} Sx && \textit{substitution } (x \neq y) \\ S &\xrightarrow{1} Sx && \textit{deletion} \\ xS &\xrightarrow{1} S && \textit{insertion} \end{aligned}$$

To frame this input/output process in a more standard fashion, one can simply assert a new starting nonterminal S', a rule $S' \rightarrow Sw\tau$ where $w \in \Sigma^*$ is the input string and τ is a new terminal marker not in the language, and an absorbing rule $S \rightarrow \tau$ that is guaranteed to complete any derivation and leave only the output string.

Note that the insertion rule is not strictly context-sensitive (the left side being longer than the right), and can generate any string whatever as output. The numbers above the arrows here represent a cost of applying the corresponding edit rule. An overall derivation would again move the S nonterminal from the beginning of an input string to the end, leaving the output to its left, as follows:

$$S\text{gact} \overset{0}{\Longrightarrow} \text{g}S\text{act} \overset{1}{\Longrightarrow} \text{gt}S\text{ct} \overset{1}{\Longrightarrow} \text{gtc}S\text{t} \overset{2}{\Longrightarrow} \text{gtcg}S\text{t} \overset{2}{\Longrightarrow} \text{gtcgt}S$$

Here the numbers above the double arrows represent the cumulative cost of the derivation. The rules applied are an identity (for no cost), a substitution of a 't' for an 'a' (adding a cost of 1), another identity, an insertion of a 'g' (adding a cost of 1), and an identity.

Minimal edit distances are typically calculated with dynamic programming algorithms that are $O(nm)$ in the lengths of the strings being compared. The same order of results can be obtained with the appropriate table-based parsers for grammars such as that above, though perhaps with the sacrifice of some efficiency for the sake of generality. The great advantage of the parsing approach is that grammars and their cognate automata make it possible to describe more complex models of string edits, and therefore of processes related to molecular evolution. The author has recast a number of the algorithms developed for such purposes in the form of automata, which can be shown to be equivalent to the recurrence relations typically used to specify such algorithms [4].

References

1. Searls, D.B.: The linguistics of DNA. Am. Sci. 80, 579–591 (1992)
2. Dong, S., Searls, D.B.: Gene structure prediction by linguistic methods. Genomics 23, 540–551 (1994)
3. Searls, D.B.: String Variable Grammar: a logic grammar formalism for DNA sequences. J. Logic Prog. 24, 73–102 (1995)
4. Searls, D.B.: Formal language theory and biological macromolecules. In: Farach-Colton, M., Roberts, F.S., Vingron, M., Waterman, M. (eds.) Mathematical Support for Molecular Biology, pp. 117–140. American Mathematical Society, Providence (1999)
5. Searls, D.B.: The language of genes. Nature 420, 211–217 (2002)
6. Chiang, D., Joshi, A.K., Searls, D.B.: Grammatical representations of macromolecular structure. J. Comp. Biol. 13, 1077–1100 (2006)