Formal Grammars

James E. Allen, 25th October 2011

Introduction

Formal grammars were devised as useful tools and models in the field of linguistics, notably by Noam Chomsky (1959), who described a hierarchy of grammars with varying levels of descriptive power (Figure 1). These transformational grammars are, however, a general theory for modelling strings of characters, and are as applicable to the 'language' of DNA and RNA as they are to English or Swahili. A grammar produces (or generates) a language through a series of productions (or rewriting rules) which define states (valid arrangements of the characters), and permissible transitions between different states. Automata of differing levels of complexity are able to recognise and parse different grammars (Figure 1). Durbin *et al.* (1998) review formal grammars with respect to biological sequence analysis.

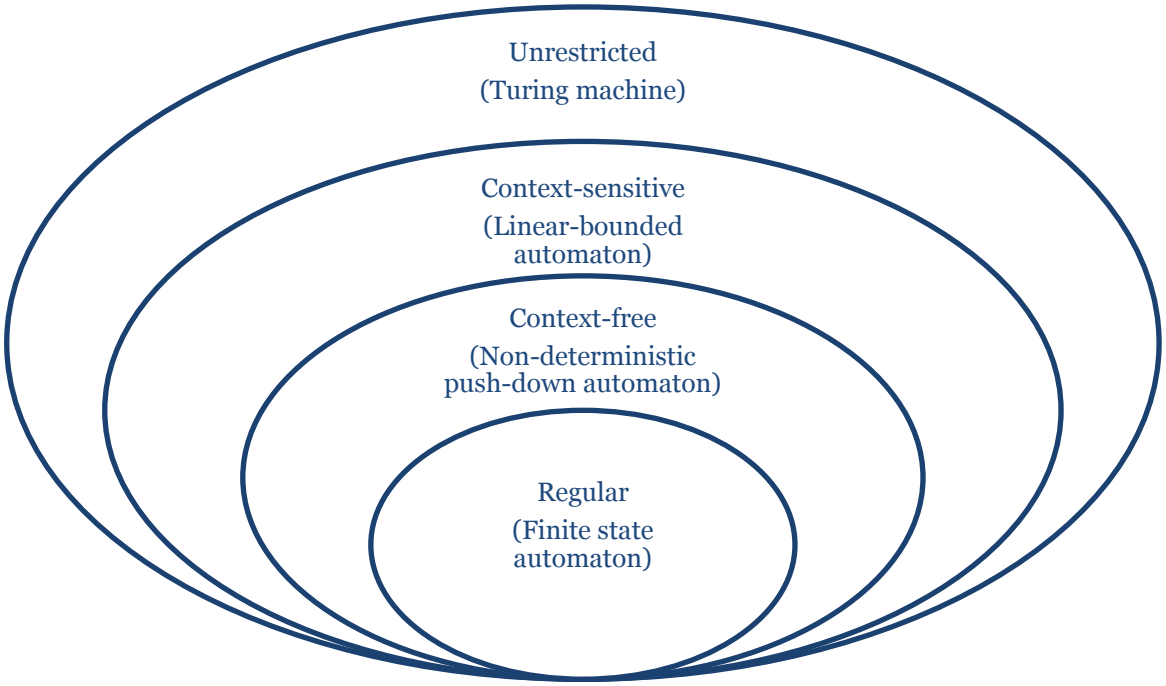

Figure 1: Chomsky Hierarchy of Transformational Grammars
The text in parentheses indicates the minimal (i.e. least complex) automaton required to model the grammar. The languages that can be generated by the grammars are given the same name as the grammar, (e.g. a regular grammar generates a regular language) except in the case of unrestricted grammars, which generate recursively enumerable languages.

Regular Grammars

The production rules in a grammar operate on symbols, of which there are two types, terminal and non-terminal; by convention these are represented by lower and upper case letters, respectively. The terminal symbols are the characters of the alphabet

(e.g. {A, C, G, U} in the case of RNA), and the non-terminal symbols denote production rules which may contain a combination of terminal and non-terminal symbols. An example of a regular grammar that generates a sequence of DNA is shown in Figure 2.

Production rules:

- $\boldsymbol{S} \rightarrow \boldsymbol{\alpha S}$
- $\boldsymbol{S} \rightarrow \boldsymbol{\alpha}$

where $\boldsymbol{S}$ is the 'Start' non-terminal symbol, and $\boldsymbol{\alpha} \in$ {A, C, G, T}.

Example:

Starting with the $\boldsymbol{S}$ symbol, we use the first rule to generate the string A$\boldsymbol{S}$ ($\boldsymbol{S} \Rightarrow$ A$\boldsymbol{S}$), applying this rule twice more (A$\boldsymbol{S} \Rightarrow$AT$\boldsymbol{S} \Rightarrow$ ATC$\boldsymbol{S}$), then using the second rule gives ATCA.

Figure 2: A Regular Grammar for DNA Sequences
Regular grammars are simple, and cannot model complex dependencies between characters, but can be generate every possible DNA sequence from two basic production rules. Only the two production rules shown are permissible in a regular grammar.

A finite state automaton (FSA) parses a string one character at a time, either accepting or rejecting the character, and if the end of the string is reached without any rejections parsing has been successful. FSAs may accept a character on transitions (Mealy machines) or on states (Moore machines); the two methods are essentially equivalent (Durbin *et al.* 1998, p.239).

Context-Free Grammars

Context-free grammars (CFGs) are required to generate languages that contain palindromes, as regular grammars cannot model the necessary dependencies between characters. CFGs have production rules that contain any combination of terminal and non-terminal symbols on the right side of the rule (but are restricted to a single non-terminal symbol on the left). Complementary base pairing in RNA is essentially a palindromic process with base pairs instead of identical characters, and can be modelled by a CFG (Figure 3).

The process of parsing a context-free language can be conceptualised as the construction of a tree, with internal nodes mapping to non-terminal symbols and leaves to terminal symbols; the children of each internal node are the symbols produced by the application of a rule (Figure 3). In this representation, all subtrees describe contiguous sections of the whole string, which makes processing with recursive algorithms quicker and easier (Durbin *et al.* 1998, p.244-5). The parsing

automaton for a CFG is a push-down automaton, which maintains a partial memory of processed states by using a stack, which can be very inefficient for non-deterministic CFGs.

Production rules:

- $S \rightarrow aSu \mid uSa \mid cSg \mid gSc \mid gSu \mid uSg$
- $S \rightarrow \alpha S$
- $S \rightarrow \alpha$

where S is the 'Start' non-terminal symbol, and $\alpha \in$ {A, C, G, U}.

Example:

A hairpin loop (below left) can be generated by three applications of the first rule, then three of the second rule, then one of the last rule: $S \Rightarrow$ ASU $\Rightarrow$ AUSGU $\Rightarrow$ AUGSCGU $\Rightarrow$ AUGASCGU $\Rightarrow$ AUGAUSCGU $\Rightarrow$ AUGAUGSCGU $\Rightarrow$ AUGAUGGCGU. This process can be represented by a parse tree (below right), where internal nodes correspond to non-terminal symbols.

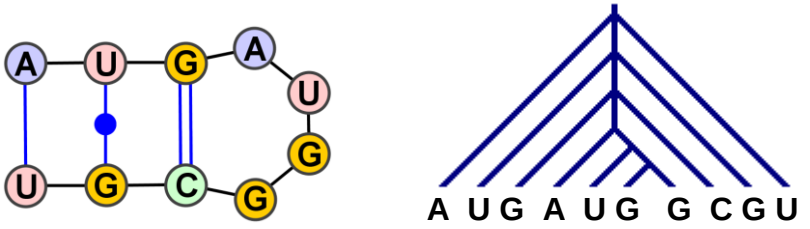

Figure 3: A Context-Free Grammar for RNA Helices and Loops
For brevity, the six rules defining complementary base pairing (including wobble pairs) are shown on one line. The image of RNA structure was created with VARNA (Darty *et al.* 2009).

Stochastic Context-Free Grammars

In a normal CFG the application of each rule is considered equally likely, but it is relatively straightforward to associate probabilities with rules, thereby creating a stochastic context-free grammar (SCFG). The probabilities of all productions from a non-terminal symbol sum to 1, and an SCFG defines a probability distribution if all possible terminal symbols have an associated probability. Stochastic regular grammars are defined in the same way, and are equivalent to hidden Markov models (HMMs), so, in a sense, SCFGs are like HMMs with a degree of memory.

When analysing biological sequences with SCFGs there are three common tasks: parsing a sequence with a given, parameterised, SCFG; calculating the probability of a sequence for a given SCFG; and estimating the parameters of an SCFG from a training set. Each problem requires a slightly different algorithm, which are outlined in Durbin *et al.* (1998, p.252-8); see also Lari and Young (1990).

Context-Sensitive and Unrestricted Grammars

Context-sensitive grammars are an extension of CFGs that allow any combination of terminal and non-terminal symbols on either side of a production rule, with the restriction that there cannot be more symbols on the left than on the right. These grammars are able to describe complex dependencies between characters, such as the nested sets of complementary base pairs that appear in RNA pseudoknots. However, the linear bounded automaton that parses context-sensitive grammars operates in exponential time, so these grammars are not in widespread use for practical reasons.

Unrestricted grammars place no limits on the number of symbols that appear on either side of a production rule. Parsing, with a Turing machine, is not guaranteed to complete (in finite time), and they are thus chiefly of theoretical interest. Stochastic versions of context-sensitive and unrestricted grammars are possible, but they are tricky to formulate, as a non-terminal symbol may have different production rules in different contexts, so the probabilities will not necessarily sum to 1.

The Language of RNA

The use of SCFGs to model the secondary structure of RNA molecules was independently proposed by Sakakibara *et al.* (1994) and Eddy and Durbin (1994), following a description of biological sequences in the framework of formal grammars (Searls 1992; see also Searls 1997). Sakakibara and colleagues take an explicitly grammar-based approach, while Eddy and Durbin develop covariance models (CMs) as an extension of the idea of profile-HMMs that are used to characterise protein families (Durbin *et al.* 1998), but the basic idea is the same in both cases.

Covariance Models

Covariance models are used in the RFAM database to define RNA families (Gardner *et al.* 2011), using the INFERNAL (Nawrocki *et al.* 2009) software to generate CMs from hand-curated 'seed' alignments of different ncRNA molecules. These CMs are then used to search for homologs in sequence data that has first been reduced to a manageable size by a BLAST search, as the algorithms used with CMs are computationally expensive (Gardner 2009). A similar approach is used by TRNASCAN-SE (Lowe and Eddy 1997) to detect tRNA in genome sequences, using fast (but rather inaccurate) programs that were specifically designed for tRNA search as an initial filtering step. Specialised programs such as these are generally not

available, but the latest version of INFERNAL incorporates filters that use HMMs and efficient algorithms to make the application of CMs far less computationally intensive (Nawrocki *et al.* 2009).

Citing this Document

[If referring to this document, please cite its location on the Monkeyshines website: http://www.monkeyshines.co.uk/blog/archives/1423]

References

Chomsky N. (1959) On certain formal properties of grammars. *Information and Control* **2**(2): 137-167.

Darty K, Denise A, Ponty Y. (2009) VARNA: Interactive drawing and editing of the RNA secondary structure. *Bioinformatics* **25**(15): 1974-1975.

Durbin R, Eddy SR, Krogh A, Mitchison G. (1998) *Biological Sequence Analysis*. Cambridge University Press, Cambridge.

Eddy SR, Durbin R. (1994) RNA sequence analysis using covariance models. *Nucleic Acids Research* **22**(11): 2079-2088.

Gardner PP. (2009) The use of covariance models to annotate RNAs in whole genomes. *Briefings in Functional Genomics & Proteomics* **8**(6): 444-450.

Gardner PP, Daub J, Tate J, Moore BL, Osuch IH, Griffiths-Jones S, Finn RD, Nawrocki EP, Kolbe DL, Eddy SR *et al.* (2011) Rfam: Wikipedia, clans and the "decimal" release. *Nucleic Acids Research* **39**(Database issue): D141-145.

Lari K, Young SJ. (1990) The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech & Language* **4**(1): 35-56.

Lowe TM, Eddy SR. (1997) tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Research* **25**(5): 955-964.

Nawrocki EP, Kolbe DL, Eddy SR. (2009) Infernal 1.0: inference of RNA alignments. *Bioinformatics* **25**(10): 1335-1337.

Sakakibara Y, Brown M, Hughey R, Mian IS, Sjölander K, Underwood RC, Haussler D. (1994) Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research* **22**(23): 5112-5120.

Searls DB. (1992) The linguistics of DNA. *American Scientist* **80**(6): 579-591.

Searls DB. (1997) Linguistic approaches to biological sequences. *Computer Applications in the Biosciences: CABIOS* **13**(4): 333-344.