To appear in Genomics

Gene Structure Prediction by Linguistic Methods

Shan Dong and David B. Searls
Department of Genetics
University of Pennsylvania School of Medicine
422 Curie Boulevard
Philadelphia, PA 19104-6145 USA

Abstract

The higher-order structure of genes and other features of biological sequences can be described by means of formal grammars. These grammars can then be used by general-purpose parsers to detect and assemble such structures by means of *syntactic pattern recognition.* We describe a grammar and parser for eukaryotic protein-encoding genes, which by some measures is as effective as current connectionist and combinatorial algorithms in predicting gene structures for sequence database entries. Parameters on the grammar rules are optimized for several different species, and mixing experiments performed to determine the degree of species specificity and the relative importance of compositional, signal-based, and syntactic components in gene prediction.

Introduction

Formal language theory views *languages* as sets of strings over some *alphabet*, and specifies potentially infinite languages with concise sets of rules called *grammars* [10]. Grammars are an exceptionally well-studied methodology, familiar to all computer scientists, for the description of complex, higher-order structures embodied in strings of symbols. Moreover, they can be given as input to general-purpose programs called *parsers* capable of determining whether a given string satisfies the rules of the grammar. Parser technology is also extensively developed, and has been applied as well to the problem of *searching* for complex patterns specified by grammars in large amounts of data, in a technique known as *syntactic pattern recognition* [6].

A formidable pattern recognition problem in biology is the recognition of protein-encoding genes in otherwise uncharacterized primary sequence data. Traditionally this has devolved to the problem of recognizing coding sequence using a variety of statistical metrics, recently reviewed in [4]. These *compositional* methods, used in what Staden termed "gene search by content" [22], typically produce for any sample window of sequence a measure of similarity, by some criterion, to "typical" exonic sequence data. Among the more commonly-used compositional measures are Fickett's TESTCODE algorithm [3], which measures *positional* *asymmetry* or the tendency for base compositions to vary systematically with position within the codon, and *hextuple frequencies* or the relative frequencies of occurrence of each 6-mer of bases in coding (either in-frame or independent of frame) versus non-coding sequences [2]. All such methods have the disadvantage that their accuracy invariably declines with smaller window sizes, and for most metrics the optimum window size is greater than the mean exon size in typical vertebrate gene sets. Nevertheless, new systems such as GRAIL have markedly improved upon these methods by combining evidence from a number of them in a connectionist architecture [24].

Another approach to gene-finding involves what Staden termed "gene search by signal" [22], the recognition of specific local binding sites or other cues to processes involved in gene expression, such as splice sites. The subtlety and degree of variation in such signals means that their detection is often as uncertain as the more global compositional measures, yet progress has also been made on this front using sophisticated statistical and machine learning techniques such as neural networks. *Weight matrices* are a widely-used and relatively "low-tech" example of a means of detecting such local signals [23]. Recently, a number of systems (including more recent versions of GRAIL) have united compositional and signal detection techniques in hybrid gene prediction systems, in which evidence is combined to predict the most likely gene structure from a stretch of primary sequence. Not only do such gene assembly programs provide more information than strictly compositional exon finders, but by imposing additional constraints they can improve the latter's performance. Several such systems have been built on rule-based architectures [5, 7]. These advances have brought into focus the combinatorial problem of assembling and testing large sets of candidate exons; this has been addressed in the GeneID system [9] by clustering exons into equivalence classes, and in the GeneParser system [21] by a novel dynamic programming approach (see also [8]). These systems achieve similar levels of performance [21].

We proposed the use of formal grammars to assemble gene structures from primary sequence in 1988 [15], and since that time have worked to build a domain-specific parser to enable the use of grammars as versatile yet reasonably efficient pattern recognition tools [16, 17, 18]. We have successfully used this system, called GenLang, for the recognition of tRNA genes, group I introns, and a variety of other features [20, 19]. We now report on more comprehensive efforts to use the GenLang system to recognize and predict structures of eukaryotic protein-encoding genes.

Methods

Grammars and Parsing

The gene grammars to be described were implemented in the logic programming language Prolog, using a powerful and extensible grammar paradigm called *definite clause grammar* (DCG) [14]. DCGs are directly translated by Prolog compilers to executable code for simple recursive-descent parsers. While DCGs offer an excellent rapid-prototyping environment and are optimized for this form of search, a number of adaptations were made in the course of developing the GenLang parser, largely for the sake of efficiency and ease of grammar development in the domain of DNA sequence data. Thus, Genlang grammars are augmented with many hidden parameters and additional functionalities, and the lower levels of the system are implemented in the 'C' programming language. One particularly important speedup is the use of *chart parsing* techniques, related to dynamic programming, in which intermediate results (i.e. parse subtrees) are saved for later re-use in highly nondeterministic parsing; chart parsing allows for an order of magnitude speedup for the grammars described below. Details of the implementation of GenLang can be found elsewhere [19, 20].

A significant feature of GenLang grammars is the incorporation of a notion of *cost*. To allow for imperfect matching, for example with a simple oligonucleotide sequence, a maximum cost may be imposed on the rule when it is invoked, and up to that number of mismatches is allowed. However, much more complicated cost models are possible, including user-defined functions, such as edit distance and weight matrix scores (see below). Costs are promulgated up the resulting parse tree and summed at each node, so that not only the overall parse can have a threshold cost, but also each subtree. This allows for rules of varying and even context-sensitive stringency.

The overall design and cost model employed for the gene parsing task was as follows. A set of rules designated as *leaf rules* was chosen, whose members referred directly to primary sequence and were adjudged to be the significant units of "evidence" for some signal or compositional measure related to the presence of a gene (enumerated in detail below). The grammar was then elaborated so that an additional set of *node rules* each invoked exactly two of either the leaf rules or other node rules. (In formal terms, the core grammar was thus reduced to *Chomsky normal form*, and it is known that *any* context-free grammar can be so structured [10].) The resulting parse trees were therefore binary. Each node rule N was responsible for combining the cost of two lower-level rules – a left (L) and right (R) child – and passing it up to its own parent. For this purpose a uniform cost function was designed based on a single "mixing" parameter, μ, as follows:

$$Cost_N = (1 - \mu) \cdot Cost_L + \mu \cdot Cost_R$$

The parameter μ was intended to range between zero, giving full weight to the left child and none to the right, and one, shifting all weight to the right child. Also associated with each node rule was a pair of thresholds, θ_L and θ_R, representing the maximum costs to be allowed over the left and right subtrees, respectively. That is, whenever in a developing parse tree a child node applied to some span of sequence accumulated a cost exceeding the threshold imposed by its parent, that child node would be said to fail. At that point, the grammar would *backtrack* or retry the child node at the next span of sequence allowed by the grammar. A parse succeeded whenever all subtrees could be assembled so as to satisfy all thresholds, including some top-level threshold applied to the tree as a whole. Even after succeeding, the grammar could be made to backtrack to find alternative answers, and in this case the predicted structure was taken to be the one with the minimum top-level cost.

Not surprisingly, the effectiveness of the resulting grammars was in large part determined by the values assigned to the mixing and threshold parameters; finding optimal values of these parameters is thus a major concern. The total cost of any given putative gene could be calculated directly from a single formula assembled from each of the cost functions, and it may be imagined that identical results to those given below could be arrived at by generating putative genes by whatever means and applying such a formula repeatedly. However, it is important to note that the application of thresholds at multiple levels greatly prunes the search space and provides much finer control over the acceptable subtrees. This fact, and other procedural aspects of the parse to be described below, permit (in fact, require) a less than exhaustive search of the possible gene structures to be performed.

Compositional and Signal Measures

As noted above, a number of compositional measures of exonic tendency have been proposed, most of which are interrelated to a greater or lesser degree [4]. We have chosen two of the best-known: in-frame hextuple frequency [2] and position asymmetry as measured by Fickett's TESTCODE algorithm [3]. Their use is described further below, and in the Discussion.

Local signals were largely detected using weight matrices, implemented in GenLang as follows. Input to a special form of grammar rule consists of a table F of numbers of occurrences of each base type in $B = \{a, c, g, t\}$ at each position $1 \leq i \leq n$ in a set of examples drawn from aligned consensus sequences of length n. This is compiled to code which evaluates a candidate sequence in a fashion optimized for rapid parsing [15]. The method that proved most effective with the signals described below was one that evaluates a sequence subarray S using a negative log likelihood function [23], as follows:

$$Cost = -\sum_{i=1}^{n} \left(\log F_{i,S[i]} - \log \left(\max_{b \in B} F_{i,b} \right) \right)$$

Thus the cost is the sum of the negative logs of the individual base position frequencies, normalized so that the most likely base in each position contributes zero cost. For certain signals, improved performance was achieved by taking into account negative examples (often done in machine learning techniques but generally not in weight matrices). This was accomplished by dividing the frequencies of the positive examples by those of the negatives, to produce a log likelihood ratio.

Gene Grammars

Protein-encoding gene grammars are the most complex we have built, and will not be given in their full detail. (Source code and documentation are available on request, and many aspects of the grammar's design have appeared in [15, 17, 18, 20].) Described below is the "core" of a new grammar designed specifically for automated optimization of gene prediction.

Leaf Rules

Leaf rules, which would be called "prelexical" in a linguistic context and are termed "sensors" in certain other gene prediction systems, are those which collect a variety of forms of evidence for assembly by the grammar. They constitute the leaves of the eventual parse tree, and are enumerated below. Frequencies for these features were compiled from gene sets to be described in a later section.

1. *Start Codon Consensus.* A weight matrix cost for the region extending from −6 bp to +6 bp from a putative start of translation, with an obligatory ATG. It proved most effective to use only positive examples, drawn from authentic starts of translation.

2. *ATG Spacing.* A cost inversely proportional to the distance from a putative start codon to the next preceding ATG in the sequence. For authentic start sites we observed that this distance is greater than the mean, due to a tendency to exclude non-start upstream ATGs in initial exons.

3. *Upstream Words.* A cost inversely proportional to the average, for each hextuple in a region up to 200 bp upstream of a putative start site, of the frequencies of occurrence of those hextuples upstream of authentic start sites.

4. *Exon Size.* For an obligatory open reading frame of length L in a putative internal exon, a cost proportional to the average exon length divided by L. Note that this penalizes exons that are "too short," but not exons longer than the mean, since of course the latter are less likely to arise by chance during parsing. Initial and final exons are not assessed a cost, since much wider size variations are observed in these.

5. *Coding Words.* For a putative coding region, a cost inversely proportional to the in-frame hextuple score. This score is calculated as the sum of the logarithms of the ratios of each hextuple's frequency of occurrence in coding versus non-coding regions, plus an offset to insure a positive result.

6. *Position Asymmetry.* For a putative coding region, a cost derived from Fickett's TESTCODE algorithm [3], but using tables of probabilities and weights calculated individually for the sets described below.

7. *Donor Consensus.* A weight matrix cost for the region extending from -3 to $+6$ from a putative splice donor, with an obligatory GT. Positive examples were confirmed donors, and negative examples were GT-containing sequences within 50 bp of confirmed donors.

8. *Donor Words.* At a putative splice donor, a cost proportional to the *Coding Words* cost for the in-frame portion of the region 50 bp upstream minus that for the region 50 bp downstream, plus an offset to insure a positive result. Such a differential is indicative of a transition from exonic to intronic sequence.

9. *Acceptor Consensus.* A weight matrix cost for the region extending from -14 to $+3$ from a putative splice acceptor, with an obligatory AG. Positive examples were confirmed acceptors, and negative examples were AG-containing sequences within 50 bp of confirmed acceptors.

10. *Acceptor Words.* At a putative splice acceptor, a cost proportional to the *Coding Words* cost for the in-frame portion of the region 50 bp downstream minus that for the region 50 bp upstream, plus an offset to insure a positive result.

11. *Exon Number.* A cost proportional to the average number of exons divided by that observed in the putative gene structure. Again, this penalizes only genes with fewer exons than the mean, rather than more, so this rule should be considered to simply establish a bias (for better or worse) toward genes with greater numbers of exons.

12. *Splice Quality.* A cost equal to the average cost of all introns in an entire putative gene, encompassing local and global measures as mixed at the level of the intron rule (see below).

13. *Coding Quality.* A cost equal to the average cost of exons in a putative gene, encompassing local and global measures as mixed at the level of the exon rule.

It should be emphasized that no effort was made to establish at the outset the relative values of each of these leaf rules in gene prediction. The costs produced by each were initially scaled to roughly the same order of magnitude, but the mixing coefficients are what determine the relative contribution of each sensor to the overall cost at each level of the parse, and the goal of the training regimen to be described below was to determine, *a posteriori*, locally optimal mixing coefficients. Thus if, for example, the bias toward larger numbers of exons in leaf rule 11 proved to be ill-founded or poorly formulated, then the cost contributed by this leaf node would presumably be devalued by the mixing coefficient (possibly to zero) in the training process.

Node Rules

The internal nodes of the grammar are represented in graphical form in Figure 1. The top-level rule *Gene* (analogous to *Sentence* in typical natural language grammars) invokes a *Translation* and a *Termination* rule, and these in turn invoke lower-level node rules, and so on down to the leaf rules. The *Remainder* rule is *recursive* in that its right child is itself; this permits it to reinvoke its left child, a rule for an *Intron/Exon* pair, as many times as necessary. Other recursive rules used in the overall grammar, such as one within the *Exon* rule that gathers codons into an open reading frame, are not included in the core grammar depicted. Also not shown is the "escape" from the recursion, a special rule for the final *Intron/Exon* pair.

Such recursion has an interesting interpretation under the cost model described above. For $\mu = 0$, the entire cost of *Remainder* is that of the left child, *Intron/Exon*; in particular, the top-level cost is just that of the first *Intron/Exon* pair in the gene. At every level of the recursion, each additional *Intron/Exon* is in effect treated in isolation and is required to meet the same cost threshold, θ_L. If, on the other hand, $\mu = 1$, only the cost of the right children will be passed up, meaning (eventually) just the final *Intron/Exon*; again, the intervening *Intron/Exon* pairs will be thresholded individually by θ_L. (In both cases θ_R can apply only to the final *Intron/Exon*, though there is an additional requirement in the former case that it not be less than θ_L.)

However, consider intermediate mixing parameter values, e.g. $\mu = 0.5$. The cost passed up at every invocation of *Remainder* would then be one-half the cost of the current *Intron/Exon*, plus one-fourth the cost of the next one, and so on. Besides the threshold θ_L on individual *Intron/Exon* pairs, the θ_R threshold will apply to all the remaining pairs, though in a decreasingly-weighted fashion. More formally, the top-level cost of a left recursion of depth n, encompassing the costs of $n-1$ left children, $Cost_{L_i}$, and that of a final right child, $Cost_R$, can be shown to be

$$Cost_N = (1-\mu) \cdot \left(\sum_{i=0}^{n-1} \mu^i \cdot Cost_{L_i} \right) + \mu^n \cdot Cost_R$$

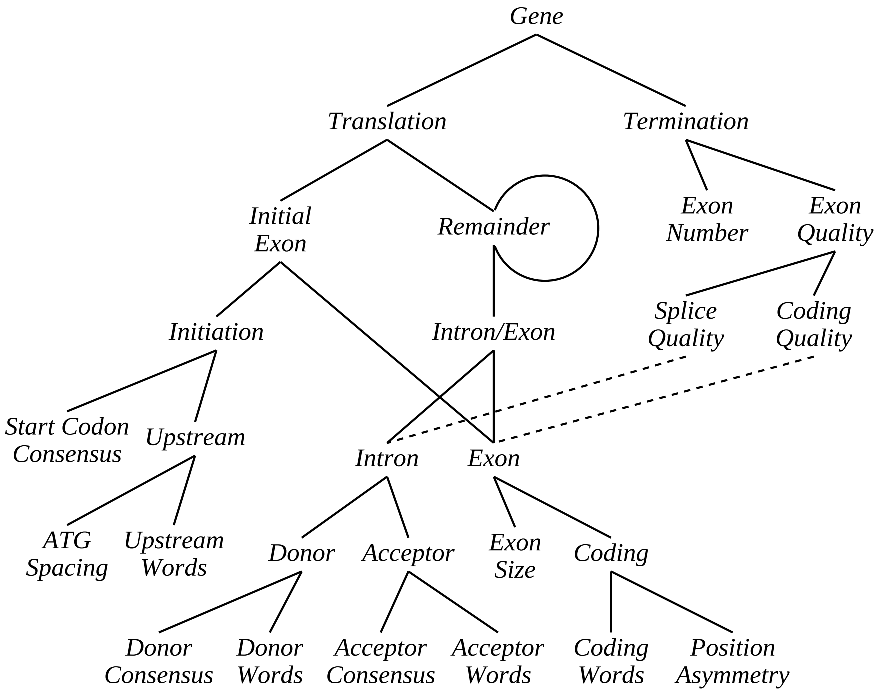

Figure 1: Graphical depiction of the core grammar structure, roughly corresponding to an instantiated parse tree of a gene. Solid lines depict rule invocations, and dashed lines indicate transmission of costs by other means. The apparent dual parentage of the *Exon* rule reflects the fact that it is invoked at different times by distinct higher-level rules. The circular arc around *Remainder* indicates recursion (see text).

This arrangement has the desirable property that the total average cost is asymptotic in the length of the recursion; more precisely, it can be shown by induction on n that, given constant $Cost_{L_i} = Cost_R = C$, it will also be the case that $Cost_N = C$ for any μ and any n. Thus, no special arrangements need be made to adjust thresholds for different numbers of exons, or at different depths of the recursion.

The rate at which the weight of subsequent terms decreases depends on μ: small μ shifts the emphasis to the left children and thus shortens the "half-life" of the costs, while larger μ shifts the emphasis to the right and levels out the contributions of all the *Intron/Exon* pairs. (In all cases, the final *Intron/Exon* contributes greater cost, the shorter the recursion; thus, the earlier the recursion is to terminate, the better the final *Intron/Exon* has to be to "justify" that termination.) The value of μ can be thought of as the effective width of a decaying "window" that determines how many *Intron/Exon* pairs are taken into account at one time by the thresholds on *Remainder*.

It should be noted that, unlike formal grammars, this grammar is not constrained to invoke terminal elements strictly in the order in which they appear on the input string. For example, it is more efficient to detect the *Start Codon Consensus* before evaluating the *Upstream* elements of the grammar. (The theory and practice of this aspect of GenLang are discussed in [16] and [18].) Moreover, while many rules refer to primary sequence directly in the tradition of formal grammars, they are intermixed with more global, evaluative rules that have a distinctly heuristic flavor. For example, the *Termination* branch of the grammar in effect constitutes a "wrap-up" evaluation of the putative gene product, in terms of the number and quality of the entire set of exons proposed. As noted in the enumeration of leaf rules, the *Splice Quality* and *Coding Quality* rules make use of the costs of the individual *Intron* and *Exon* rule invocations, respectively; however, the rules under *Termination* calculate a uniform average as opposed to the half-life window employed under the recursive rule. Particularly for more complex genes, it may be important to allow for such global as well as local assessments of exon quality.

This departure from formal grammars, and the incorporation of features such as costs, give the resulting system the flavor of a hierarchical rule-based system such as GeneID [9]. Even traditional linguistic techniques such as chart parsing have their analogues in other technical domains. However, the lower levels of the gene grammar and indeed the recognition engine itself are still general purpose tools, which have been used to find a variety of other types of features [19]. The logic grammar framework is well-suited for rapid prototyping, and inherently supports useful features such as input management behind the scenes, parameter hiding, easily modified syntax and higher level language features, and most of all an efficient backtracking search mechanism. We feel it is also very important that the practical grammar used here is still built upon and tied to a formally well-founded "idealized" gene grammar [17], which continues to provide a solid foundation for other ongoing research in parallelization, formal grammars, etc.

Gene Parsing

A single parsing run actually entails the generation and evaluation of a large number of alternative parses, with the minimum cost parse being selected as the final answer. The stringency of the thresholds in the grammar is gradually relaxed in the course of parsing. That is, while mixing coefficients μ are specified by the user before parsing begins and remain constant, the thresholds θ increase during the parse, across a range that is determined by a training set of known sample genes.

Given such a training set, the grammar is first used to parse each gene in such a way that only the known, authentic structure is allowed. (One of the advantages of the DCG methodology is that properly-written grammars may be given an *instantiated* parse tree as input, rather than producing it as output, so as to enforce a given parse [16].) Thresholds may then be set to the maximum costs encountered at each node for the entire training set, i.e. so as to just barely allow every known gene and element of a gene; this is called the 100% *setpoint* for the grammar thresholds as a group. In order to vary thresholds from these trained values, the user does not change individual thresholds but rather the setpoint for the grammar as a whole. For example, a setpoint of 120% would uniformly increase every threshold to 1.2 times the maximum value encountered in the training set.

However, setpoints less than 100% are handled differently. First, thresholds for elements occurring only once in a parse tree, such as start codons, are not allowed to decrease from the 100% value, but remain fixed at that level even for lower setpoints. For elements occurring more than once per gene, such as exons, the complete list of such costs encountered in the training set is examined. The setpoint determines what percentage of those costs are included, e.g. the threshold for exons is chosen so that the setpoint percentage of known exons from the training set would succeed.

This process is repeated for a series of setpoints, typically 80, 90, 95, 98, 99, 100, 105, 110, 120, and 130%, to determine a range of threshold values tailored to the training set. This process is called *threshold calibration* on the given set of sample genes. Note that a threshold for the *Gene* rule is also determined, though this rule is not invoked by other rules but only at the top level by the parser.

After threshold calibration, and with each mixing parameter μ having been set by a procedure to be described, the parse can proceed. For each setpoint level in order of increasing percentage (i.e., decreasing stringency), the top level rule is invoked for a given input – generally an entire GenBank entry sequence. The Prolog-based parser implements a top-down, backtracking search according to the grammar, and wherever a complete candidate gene is found compares it to the GenBank annotations and records various measures of the "quality" of the result (see below). The parser immediately backtracks to produce as many alternative results as possible, until either a specified maximum number of parses is reached (typically 100) or a time limit is exceeded (typically 2 minutes on a SPARCstation10). At this point, the next higher percentage setpoint is tried, except that the top-level threshold is now decreased to the maximum cost already encountered at the higher stringency. When either all setpoints have been tried, or a timeout has occurred at some setpoint with no parses having been produced, the parse terminates, and in batch mode the parser moves on to the next sequence in the database.

This protocol proved effective over a wide range of genes, with minimum-cost parses for different genes distributed among all setpoints. The 80% level parsing generally proceeded quickly due to the higher stringency, and where parses were found established a lower top-level threshold that in turn sped up the subsequent lower stringency (90%) parsing, and so on. This coarse-grained branch-and-bound technique, as well as the chart parsing described above, was necessary to produce large numbers of alternative structures under the inherently costly parsing methodology, though the number examined are still small compared to other combinatorial algorithms which perform exhaustive searches [9, 21].

Training and Test Sets

Our philosophy in selecting and normalizing training sets of sequence database entries was to do this in as automated a fashion as practicable, in part so as to help drive out human bias, but primarily to permit the uniform and rapid treatment of multiple species (and eventually other subdivisions of the data). Consequently, rather than using hand-picked training sets, GenLang scripts were written to select various data sets directly from GenBank. The current version of Genlang uses a DCG of GenBank entry syntax to scan the flat file versions of the database, performing selections on and extracting various header fields of interest. In addition to human entries, mouse and drosophila entries were selected (genus 'Mus' and 'Drosophila', respectively), and a much broader collection of dicot plants (class 'Magnoliopsida').

The sets of entries extracted for each of these phylogenetic groups are as follows:

A. *cDNA Gene Set.* This set contained entries selected as being of type 'm-RNA' and having the substring 'complete cds' in their description field. The coding sequence extracted from these entries can thus be expected to be full length. These sequences were used to compile statistics for start codon weight matrices, coding position asymmetry coefficients, and in-frame coding hextuple frequency tables.

B. *Genomic Gene Set.* This set consisted of entries of type 'ds-DNA', of length greater than 1000, and with a CDS annotation listing more than one coding region in a single transcript, i.e. containing at least one intron. These complete and partial genomic sequences were used to compile splice junction weight matrices, non-coding position asymmetry coefficients, and non-coding hextuple frequency tables.

C. *Complete Gene Set.* This set, essentially a subset of the genomic gene set, consisted of entries of type 'ds-DNA' having the substring 'complete cds' in their description field. These complete genomic DNA sequences were used to assess the average number of exons per gene and average internal exon length, and to compile hextuple frequency tables for regions up to 200 bps upstream of the start codons (since only in 'complete cds' entries could the beginning of the CDS list be trusted to be the actual start codon). This region can be expected to contain hextuple entries including known proximal promoter elements such as TATA boxes; this was confirmed by direct examination of the frequency tables.

D. *Parsing Gene Set.* To serve as a pool for training to establish μ and θ values, and for testing the gene grammar, 48 entries from the complete gene set were selected at random, with the following constraints:

- Only sequences less than 20,000 bp were selected. As a practical matter, this permitted a more uniform depth of parsing for each gene in the training set within the time limits set, and in any case few current entries exceed this length. Parsing of longer genes will be addressed below.
- Sequences with unknown bases were not selected. Some sensors were not designed to deal with unknown bases, and while many other algorithms substitute bases at random in such cases, we found that simply skipping them eliminated only a small percentage of complete gene entries and greatly simplified the interpretation.
- Entries exhibiting multiple gene products, for example due to known alternative splicing, were eliminated. This was accomplished by selecting only entries with a single CDS feature. Like longer gene entries, this class is growing in size and contains some of the more "interesting" gene structures, but the evaluation of performance (see below) is problematic in these cases. Parsing of alternatively spliced genes will be discussed below.

These 48 entries were removed from the genomic as well as the complete gene set, in both cases before the statistics indicated above were compiled. Moreover, members of the cDNA gene set with BLAST scores exceeding 1200 against any member of the parsing gene set were removed; this served to exclude cDNAs of the same gene as well as closely related genes in a family. The same criterion was applied to ensure that no two closely-related genes were included in the parsing gene set itself.

E. *Training/Test Sets.* The parsing gene set was divided randomly (except as noted below) into sets of 16 for three-fold cross-validation during training of the gene grammar.

That is, three training runs were performed, each time using 32 entries for training and leaving out 16 entries for testing, with different non-overlapping sets of 16 omitted in each run.

In addition to removing the parsing gene set from the cDNA, genomic, and complete gene sets, the latter were each normalized to give approximately equal consideration to every closely-related gene family, as opposed to each individual gene. For example, so as not to give undue weight to the large number of globin gene entries in sequence databases, it is desirable to cluster these into a single representative class, as in [9]. This was done by the simple expedient of counting the number of BLAST scores greater than 1200 for each entry compared against each of the other entries in the set, and assigning a weight to that entry equal to the inverse of the number of 'hits'. Then, in compiling frequency tables, weight matrices, etc., each sequence contributed its weight, instead of unity, to the counts. It can be seen that every member of a closely-related family of genes would thus contribute a weight equal to one over the size of the family, so that the entire family would possess unit weight. By the same token, unique genes would also have unit weight. Unlike the case with hand-picked representative training sets, each member of a cluster thus has a 'vote' in the overall consensus. We find that this technique used with a threshold of 1200 produces clusters that, upon subjective examination, contain nearly all obviously related entries without including a large number of unexpected associations or more distant evolutionary relationships. Note that equal weighting for each gene family actually may not be the best policy for *de novo* discovery of genes, which might be expected to occur in proportion to the current distribution of gene classes in the databases. However, for purposes of validation of the technique, it is important to minimize "crosstalk" between the training and test sets, such as might occur if the latter happened to contain genes bearing more remote similarities to large families that were heavily represented in the training set.

The sizes of the training sets and results of normalization are summarized in Table I. (The sharp reductions observed upon clustering drosophila genomic and complete gene sets are due to the very large number of alcohol dehydrogenase genes represented.) The complete parsing sets may be obtained by anonymous FTP from cbil.humgen.upenn.edu, in the directory /pub/genlang/testsets.

Table I: Effective sizes of data sets employed (number of entries).

Gene Set	Stage	Human	Mouse	Drosophila	Dicot
cDNA	initial selection	1663	813	208	235
	less parsing set	1598	784	203	210
	after clustering	1028	557	168	158
genomic	initial selection	325	179	336	153
	less parsing set	277	131	288	105
	after clustering	187	87	103	77
complete	initial selection	223	115	146	122
	less parsing set	175	67	98	74
	after clustering	130	55	35	50

Performance Assessment

A variety of means have been used to assess the quality of gene structure predictions, which we classify into the following groups:

A. *Correct Genes.* This method simply counts the number of genes correctly predicted in the entirety of their coding sequence, and expresses this as a fraction of the total number of genes attempted. This is the most stringent test and no existing software has produced very high scores.

B. *Correct Exons.* This method counts the number of exons correctly predicted from end to end, and again gives the fraction of the total. It may also be useful to count "half right" exons, or more explicitly to determine the number of splice sites of either type correctly predicted, as well as start and stop codons; note that this is different from assessing these individual signal sensors in isolation, since they are here required to be correct in some wider context.

C. *Exon Overlap.* This method determines the degree to which the set of predicted exons overlaps the authentic exons. This is done by counting the number of bases correctly predicted as exonic (i.e. true positives, TP), falsely predicted (FP), correctly excluded (TN), and falsely excluded (FN), and then applying a variety of statistical metrics, such as sensitivity and specificity.

We measure exon overlap using the following definitions:

$$\mathit{Sensitivity} = \frac{TP}{TP+FN} \qquad \mathit{Specificity} = \frac{TP}{TP+FP}$$

$$\mathit{Correlation\ Coefficient} = \frac{(TP \cdot TN - FP \cdot FN)}{\sqrt{(TP+FP)(FP+TN)(TP+FN)(FP+FN)}}$$

Note that definitions vary among authors. In particular, our formula for specificity is that used by [21]; it is called 'Sen2' by [9], and appears elsewhere in the literature as *positive predictive value* [12]. (A different formula is often used for specificity, which counts true negatives, but these numbers are generally too large in this application to be informative; in nearly every case this formula produced values that varied only between 0.95 and 0.98.) There appears to be a consensus that Matthews' correlation coefficient (CC) [13] is the best overall indicator of overlap, providing as it does a single scalar metric of performance. For all sequence entries of a given set, individual and average values for each of these metrics for the minimum cost parse were compiled. For purposes of comparison, two different methods of aggregating results were used, to be described in Results.

Training Regimen

As noted above, training of grammar parameters proceeded by way of a three-fold cross-validation using training sets of 32 entries and test sets of 16. The following regimen was

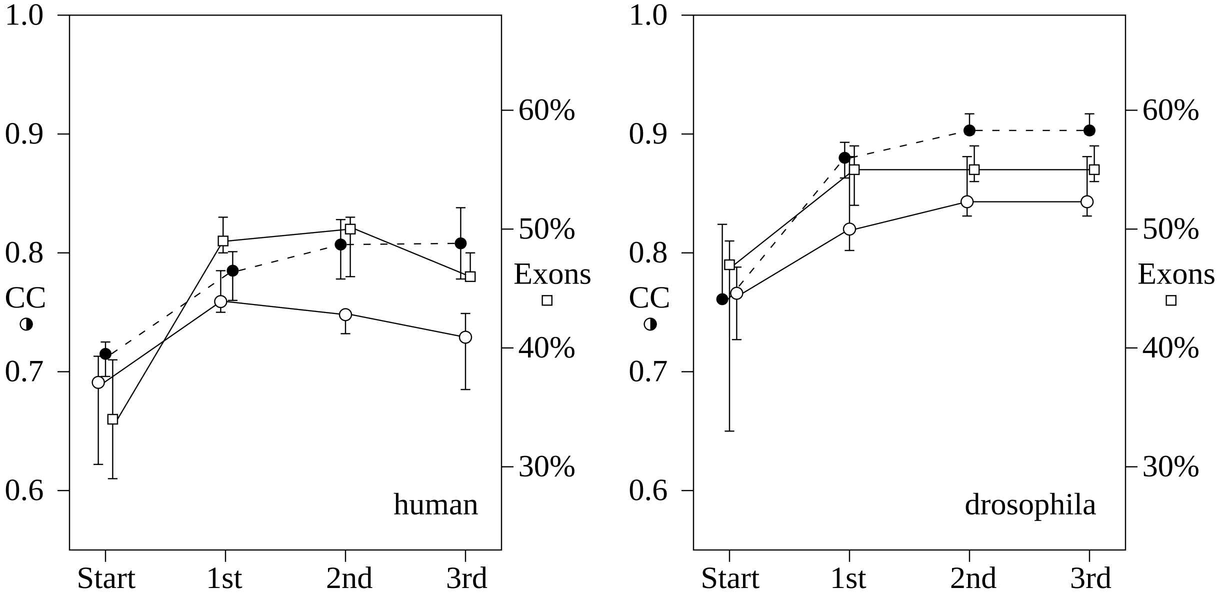

Figure 2: Training of the gene grammar on two of the four data sets (mouse and dicot not shown). Connected points are median values, while error bars indicate the remaining values, for the three cross-validated test sets. Circles give values for correlation (i.e., exon overlap) using the scales on the left, while squares indicate exon fraction correct (in percent) using the scales on the right. Filled circles and dashed lines indicate training set results, while open figures and solid lines give test set results. The horizontal axes indicate the progression of the training through three epochs, corresponding to perturbations of 0.3, 0.2, and 0.1, respectively, from the starting mixing coefficients of 0.5.

arrived at after testing a variety of "hill-climbing" schemes for adjusting parameters to a local optimum.

Training began with all mixing parameters set to $\mu = 0.5$; threshold parameters were arrived at by thresholding on the training set itself at each stage. (The same thresholds were used subsequently for the test sets.) Each training epoch consisted of a *perturbation* step followed by a *combination* step. For the perturbation step, a series of parses of the entire training set were run in which μ_N for each node N in the grammar was changed both up and down by some value Δ, with the parameters on all other nodes being held constant. Of all the results, the changes producing the five highest average CCs were collected for the combination step. In this step, all possible combinations of the five changes were tried together, and the combination producing the highest average CC was selected as the outcome of the training epoch. These changes, with all other node parameters left at their previous values, constituted the baseline for the next epoch. A total of three such epochs were run, with values of Δ equal to 0.3, 0.2, and 0.1, respectively.

This *ad hoc* training regimen, while not formally justified on probabilistic or other theoretical grounds, was adjudged the most effective of a number of alternatives tried. The alternation of perturbation and combination steps was found to be particularly important, as an earlier attempt that used only perturbation revealed that some favorable changes in distinct mixing coefficients were "self-cancelling" in combination. The amount of time required by the parsing methodology is limiting in a number of respects, including the sizes of the training and test sets, the degree of cross-validation, and most importantly, the number of different vectors of parameters that could be independently tested. However, as will be seen in the next section, training appeared to progress rapidly and effectively even under the limitations of the architecture.

Results

Grammar Training

Some results of grammar training are illustrated in Figure 2. In all cases, training sets showed sharp improvements after the initial epoch, and generally much less improvement thereafter. Human test sets started at an unexpectedly high level in terms of both correlation coefficient and exon fraction correct, showed improvement in the initial epoch, but then declined with further training; this suggested that the grammar was *overtraining,* or adapting to the specific training sets in ways that did not generalize well to the test sets. The mouse test sets (not shown) started at a somewhat lower level of performance and demonstrated continued increasing average performance through three epochs, eventually achieving correlation coefficients similar to the human peak. Drosophila and dicot test sets improved through two epochs, with drosophila sets starting at the highest levels overall. Both these sets achieved higher average performance than did those from vertebrates. We believe that this is primarily because to a first approximation vertebrates have short exons and long introns, while the opposite is the case for drosophila and dicots; for this reason global signals are stronger in the latter.

Figure 3 gives some indication of the nature of the grammars at the end of training. The parse trees shown for representative data sets are depicted in such a way that the width of each pair of arcs from parent nodes to children reflects the relative cost contribution of those children to the parent's overall cost, as determined by the mixing coefficient at the parent using input costs from the children at their 95% threshold values; these threshold values are chosen as being representative of critical cutoffs during the parses, but essentially the same results are observed when average input costs are used. Also, the grayscale intensity of the arcs is an indication of the "criticality" of the mixing coefficient of the parent node, as determined by the deviation observed in the correlation coefficient with the perturbation of that μ in the final training epoch. That is, the correlation coefficients for $\mu+0.1$ and $\mu-0.1$ are compared with that for μ at its penultimate trained value, by taking the square root of the sum of the squares of their differences from the correlation coefficient for μ.

The combination of width and intensity of these arcs gives some indication of the flow of costs up the parse tree. Some extreme differences in cost weighting between children, such as the width disparities below the *Initial Exon*, *Upstream*, and *Exon* nodes, can be somewhat discounted in importance because of the generally faint criticalities of the arcs; this indicates little sensitivity to change in μ at these nodes. It is important to note, in addition, that small weight or criticality for an arc does *not* indicate that the child node is unimportant to the result, but rather that the cost of that child need not be evaluated in combination with other costs at a higher level, or in the overall cost. Such a node may constitute strong

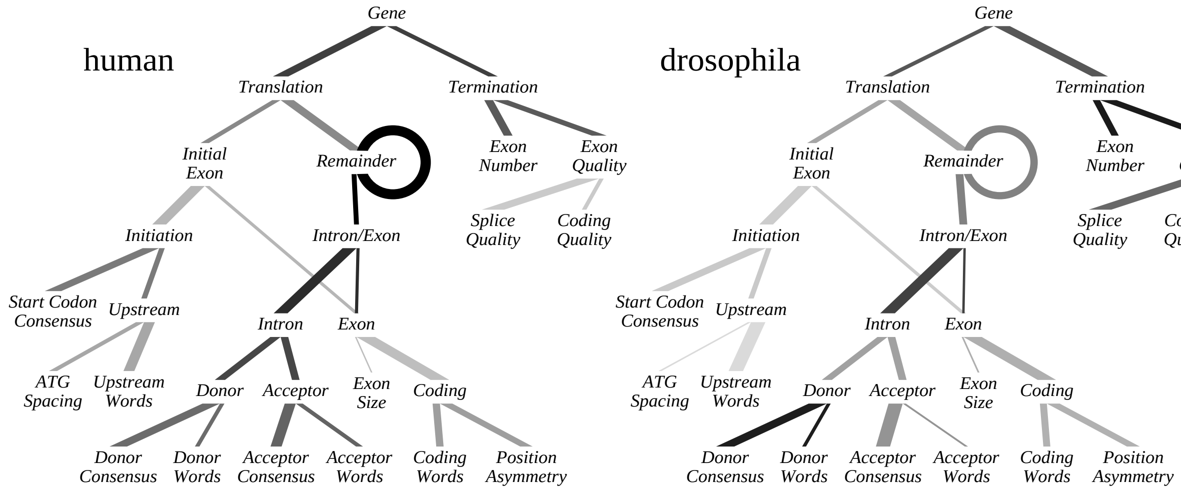

Figure 3: Nature of the grammars resulting from training on the data sets of Figure 2. The trees represent grammars as in Figure 1, and the arcs indicate status after the third epoch. Width of each arc indicates relative weight accorded that arc by the trained mixing coefficient, and grayscale intensity indicates "criticality" of the mixing coefficient to the final result, as defined in the text.

independent evidence which discriminates well based on its own local threshold, such that there is simply no great advantage to passing its cost further up the parse tree.

Values of these parameters are given as the means for the three training sets. The resulting grammars demonstrated some degree of intra-species variation in the final mixing coefficients among the three cross-validated training runs, perhaps greater than might be expected given the relative uniformity of the results of Figure 2 (see Discussion). Several other striking differences in relative weights of mixing, such as between *Consensus* and *Words* components of splice *Donor* and *Acceptor* rules, are however of only moderate criticality in all but a few instances, and these criticalities also tend to be inconsistent among training runs.

Among the more consistent observations, however, was the greater weight put on the *Translation* cost in human and mouse at the top level, as compared to the *Termination* cost which was emphasized more in drosophila and dicot. Recall that the *Translation* rule is an "on-line" procedural rule that assesses the developing cost of the gene on the fly, while the *Termination* rule is an "off-line" *post hoc* evaluation of an overall gene; the results observed suggest that both components are indeed important in combination, and in varying ratios, for the effective evaluation of the overall *Gene*. The *Termination* node itself also shows fairly consistent criticality, suggesting that the bias toward greater *Exon Number* built into the left child is effective in balance with an overall indication of *Exon Quality*.

Within the *Translation* subtree, the vertebrate datasets showed a consistent weighting and criticality at the recursive *Remainder* node, indicating that, for species in which this subtree is given greatest weight, the effective window in which *Intron/Exon* pairs are evaluated, as described in the Methods section, is an important parameter. Beneath the *Intron/Exon* node, the *Intron* rule is given somewhat greater weight than the *Exon* rule, and the criticality suggests that in the animal datasets it is fruitful to threshold them in combination at this particular ratio.

Every effort was made to ensure that test sets were properly separated from training sets, as noted in the Methods section. Not only the removal of training sequences showing a high similarity score with test set sequences, but also the clustering of highly similar entries in the training sets, should have helped to avoid any undue influences. In addition, we believe that this is the first cross-validated study in the arena of gene prediction. In order to ensure that more subtle similarities within the final training sets were not affecting test results, we looked for any correlation between performance and the cumulative BLAST scores between individual test sequences and the training set as a whole. We found no suggestion of any such correlation, measuring performance either by exon overlap or exon fraction correct (data not shown).

Comparative Studies

These overall results appear to be similar to those achieved with a variety of other special-purpose gene assembly algorithms, even though the GenLang parser is a general-purpose pattern matching tool not especially adapted to exhaustive combinatorial search. Direct comparisons are difficult, for a number of reasons:

- Test sets are not well standardized, and even comparisons on test sets that have appeared previously must generally use slight variations because of faults identified in entries, changes in the underlying databases, or individual limits or requirements of the systems under test.

- Different metrics and standards are employed by different workers. We have described some of the differences in performance measures, but other authors use completely different means of evaluation. Moreover, it is difficult to adjust for differences between authors in separation of training and test sets, normalization of training sets to adjust for clusters of similar genes, and cross-validation.

- Systems are generally distinct by virtue of their overall architectures, their selections of sensors, their means of combining evidence, and their approach to the combinatorics of exon assembly, so it is difficult to attribute performance differences to any particular factor.

- Gene prediction is a very active field of research, and systems are constantly being improved, so that any comparative results are likely to have been "leapfrogged" by the time they reach press.

With these caveats in mind, we nevertheless compared GenLang's performance with several other programs in current use. Table II shows the results of such a comparison with a recent version of GRAIL (XGRAIL, October 1993, obtained by anonymous FTP from arthur.epm.ornl.gov). Statistics from individual entries are averaged in two ways: by gene, as in [9], and by base (i.e. by summing results on a nucleotide by nucleotide basis), as in [21]. Thus, for example, for a test set of n sequence entries the formula for sensitivity by gene would be $(\sum_{i=1}^{n}(\mathrm{TP}_i/(\mathrm{TP}_i + \mathrm{FN}_i)))/n$, while that for sensitivity by base would be $(\sum_{i=1}^{n} \mathrm{TP}_i)/\sum_{i=1}^{n}(\mathrm{TP}_i + \mathrm{FN}_i)$. We use the former method in presenting our other data,

Table II: Comparison with GRAIL using GenLang Test Sets.*

	Test			by gene			by base		
Program	Set	Gene	Exon	CC	Sn	Sp	CC	Sn	Sp
GenLang	1	.06	.49	.76	.86	.78	.74	.84	.72
	2	.19	.50	.79	.85	.82	.77	.83	.79
	3	.13	.52	.75	.79	.82	.74	.77	.81
	mean	.13	.50	.77	.83	.81	.75	.81	.77
GRAIL	1	.00	.40	.72	.73	.83	.78	.78	.85
	2	.19	.47	.81	.81	.86	.85	.85	.91
	3	.37	.62	.81	.76	.94	.85	.78	.97
	mean	.19	.50	.78	.77	.88	.83	.80	.91

* Data shown are Gene and Exon fractions correct, and statistics for correlation (CC), sensitivity (Sn), and specificity (Sp) calculated by gene or by base for human test sets, as described in the text.

because it gives a better indication of the success rate over each attempted sequence entry, and avoids lending extra weight to genes with longer, "easier" exons. The results in Table II illustrate why the distinction is important: in terms of overall correlation the two programs are comparable when measured gene by gene, but GRAIL performs better when averaged by base. We believe that this is because GRAIL does slightly better on genes with larger numbers of exons, and GenLang is better at recognizing exons less than 100 bases in length (data not shown); both these factors would tend to favor GRAIL when performance is measured base by base. (A more recent version of GRAIL achieves improved performance on smaller exons in part by allowing for assessment of coding measures over variable window sizes [E. Uberbacher, personal communication], which is inherent in the GenLang system.)

Table II indicates that, as a rule, GRAIL performs with higher specificity, while GenLang is relatively more sensitive. Reports of GRAIL results generally emphasize specificity as a measure of performance [24], probably reflecting an orientation toward the use of GRAIL to predict putative exons for further laboratory investigation, a situation in which false positives can prove very costly. Figure 4 shows a direct comparison of the two systems for each gene attempted; the degree of scatter between the individual results of the two programs indicates that to some extent they have disjoint strengths and weaknesses, and that in general it may be advisable to perform multiple analyses of novel sequence data.

We also analyzed the performance of GenLang relative to two other special-purpose gene prediction programs, GeneID [9] and GeneParser [21], based on data from these original papers using the GeneID test set. To do this we retrained the GenLang grammar using 35 entries from our training/test sets which did not appear in this new test set (also removing from our larger sets all entries similar to any member of the new test set, by the criteria used previously). Based on the training results above, only a single training epoch was used, and it was again confirmed that this was sufficient to achieve peak performance. Table III compares these results with original data (recalculated as required for direct comparison) from [9] and [21].

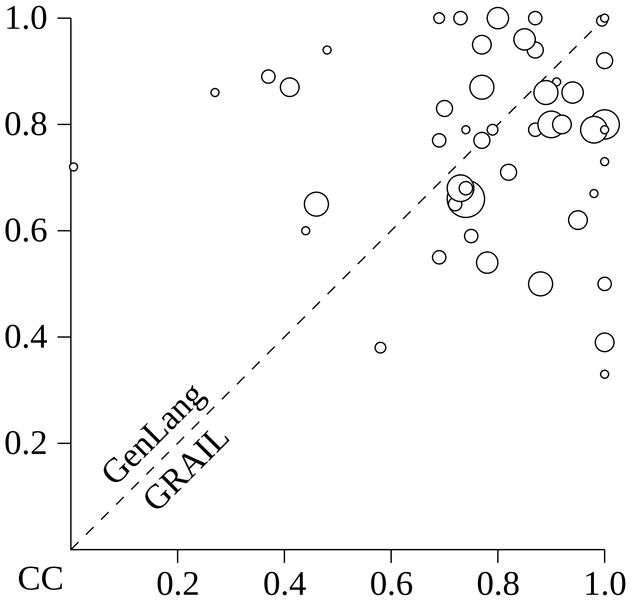

Figure 4: Performance of GRAIL vs. GenLang on individual entries from the GenLang human test sets. Each circle represents a single entry, with correlation scores from the two programs indicated by positions on the respective axes. Thus, points above and to the left of the dashed line represent entries for which GenLang's correlation score exceeded that of GRAIL. The radius of each circle is proportional to the number of exons in the gene, ranging from three to 14.

Once again, GenLang demonstrates generally greater sensitivity and lower specificity than the other programs, and comparable overall correlation scores. The GeneID test set contained slightly less than half human sequences, the remainder being mostly rodent. Since GenLang showed lower correlation using this test set than it did with its cross-validation test sets, the human subset (consisting of 12 sequences) was extracted and tested separately. Indeed, GenLang's performance improved in all respects and was more consistent with the human-only study above. At the same time, results with GeneID itself (whose training set was of mixed species composition, similar to the test set) did not show such improvement.

Because the grammars were trained to maximize correlation, which essentially represents a tradeoff between sensitivity and specificity, there would presumably be some degree of freedom on this axis. We also tried training GenLang grammars using maximization of specificity as the selection criterion. Table III shows that this change did indeed result in higher values for specificity, but at the expense of both sensitivity and correlation, and not to the specificity levels achieved by GRAIL in the data of Table II. Training for maximization of exon fraction produced identical results to those seen with correlation training (data not shown).

Phylogenetic Specificity

The data of Table III suggests that performance may have been enhanced by training and testing in a species-specific manner. To test this we tried the individual grammars on each heterologous phylogenetic group, with results shown in Figure 5.

Table III: Comparison with GeneID and GeneParser using GeneID Test Sets.*

	Test			by gene			by base		
Program	Set	Gene	Exon	CC	Sn	Sp	CC	Sn	Sp
GenLang	total	.07	.47	.69	.77	.73	.72	.78	.74
	human	.17	.52	.76	.82	.77	.79	.85	.79
GeneID	total	.11	.44	.67	.69	.84	.67	.65	.78
	human	.17	.46	.66	.72	.80	.69	.70	.80
GeneParser	total	—	.49	—	—	—	.69	.68	.78
GenLang	total	.04	.26	.60	.55	.83	.58	.46	.85
(Sp-trained)	human	.08	.37	.69	.61	.89	.65	.55	.87

* The total test set was the mixed vertebrate set used by [9] but excluding the HAMRPS14A entry (as in [21]), for a total of 27 entries; the human test set consisted of the 12 human sequences in the total test set. Data were as in Table II, and were taken from [21] for GeneParser and from [9] for GeneID, in the latter case by recalculation from the original entries for the human subset and for the statistics by base.

These results indicate that there are indeed differences among the grammars, although the differences between the human and mouse grammars are slight and not likely to be statistically significant, based on the variation observed in the cross-validation. For example, the human-trained grammar gave a correlation of 0.75 on a human test set, 0.72 on mouse, but only 0.27 on dicot plants. Similarly, the dicot-trained grammar produced a correlation of only 0.21 on the human test set. Performances for exon fraction showed similar patterns, and with a few exceptions the matrices are roughly symmetric. In order to summarize cross-species differences we calculated an *ad hoc* distance between each pair of phylogenetic groups, defined for correlation as $(\mathrm{CC}_{1,1} \cdot \mathrm{CC}_{2,2}) - (\mathrm{CC}_{1,2} \cdot \mathrm{CC}_{2,1})$, where $\mathrm{CC}_{1,2}$ indicates the individual correlation between training set 1 and test set 2, etc. (and similarly for exon fraction).

In an attempt to dissect out the reasons for the apparent phylogenetic differences in the grammars, we performed mixing experiments in which various compositional and signal measures, as well as grammar parameters, were combined from and applied to different species. Thus, for example, the human-trained grammar from Figure 5, which demonstrated a correlation of 0.75 and exon fraction of 0.51, was tested with drosophila compositional measures (hextuple tables and position asymmetry frequencies) substituted for the human ones; this reduced the performance metrics to 0.47 and 0.26, respectively. When the human grammar was used with drosophila local signals (weight matrices for splice junctions, etc.), the effect was less deleterious, with performance reduced to 0.70 correlation and 0.37 exon fraction. Similar results were obtained when drosophila mixing coefficients, governing the combination of evidence, were substituted in the human grammar: 0.66 and 0.35, respectively. These mixing experiments were tried for all combinations of species (data not shown) with essentially analogous results, though varying in overall degree. To summarize, compositional measures were the most important factors in performance, but there were indications that all aspects of the grammars contribute in some degree.

Correlation

train \ test	H	R	D	P
H	.75	.72	.31	.27
R	.75	.77	.63	.55
D	.33	.36	.83	.47
P	.21	.23	.61	.87

Exon Fraction

train \ test	H	R	D	P
H	.51	.52	.17	.16
R	.48	.55	.28	.26
D	.10	.14	.55	.26
P	.05	.06	.35	.67

Distance

exon fraction \ correlation	H	R	D	P
H		.04	.52	.60
R	.03		.41	.54
D	.26	.26		.44
P	.33	.35	.28	

Figure 5: Application of grammars to heterologous phylogenetic groups. Row and column labels indicate phylogenetic groups of training and test sets employed, where H=human, R=mouse, D=drosophila, and P=dicot plants. The first two matrices give results for correlation and exon fraction, while the last matrix shows correlation and exon fraction distances, as defined in the text. Mixing coefficients and test sets used were selected from the previous cross-validation experiment based on the results which gave the highest combined performance on training and test sets, and then applied consistently across other species.

Discussion

In general the parser was surprisingly robust in the face of varying vectors of mixing coefficients. Direct examination of the trained coefficients suggested that only a few key changes were required in any species to achieve most of the results observed, and most individual perturbations produced a neutral or only slightly detrimental effect. In fact few generalizations were possible concerning the optimization process. In each species the three training sets did not always appear to be converging on the same vector of coefficients, even though final results were comparable, suggesting that in this system the surface of optimal vectors may be relatively shallow and "corrugated" with many local optima. Indeed, when training was performed on one species starting with coefficients that had been developed by training on another species, the final vector of coefficients had characteristics of both the species' final vectors under the usual training, and the final performance was somewhat degraded (data not shown).

There is no guarantee that we have arrived at global or near global optima under the *ad hoc* training regimen used, and the observations above suggest that there may be many suboptimal local cost minima to which it is susceptible. Possibly a more effective training methodology would result in even better performance, but the length of time required to test each change by parsing limits the amount of search that can be accomplished with the present technique. (The results of this paper required on the order of three CPU-months of computer time.) We are currently investigating means by which this process may be sped up to allow more extensive search, perhaps by analyzing parse trees to allow credit assignment at a more detailed level (short of complete parses) for backpropogation, as well with as alternative search methodologies such as genetic algorithms. Any improvement in efficiency would allow us to increase the number of training examples, which at 32 is not large compared to the number of parameters (14 mixing coefficients), though it should be noted that there were typically 100-200 exons in the training sets for the critical internal nodes.

On the other hand, the overtraining observed and the variation among test sets in final mixing coefficient vectors suggests that additional training may produce diminishing returns. This is supported also by the observation that, during training, improvements in one subset of genes would typically produce offsetting losses in another subset, even after several epochs. This suggests the possibility of disjunctive grammars that essentially subclassify genes, by "branching" the training process. In effect, something like this is already being done by way of the sliding threshold values, which on some genes produce many alternatives at the stringent earlier thresholds, and on others produce no parses at all until the thresholds are greatly relaxed. Others have observed a "feast or famine" effect whereby sequence entries vary tremendously in the numbers of potential genes they produce (ranging continuously over five orders of magnitude in [9]).

It would of course be most interesting if such disjunctive grammars actually recognized functional differences in the "types" of genes that they recognized, for instance if they happened to classify genes by time or tissue of expression, perhaps reflecting subtle differences in the gene expression machinery. The apparent species specificities we have identified may in fact reflect nothing more than selection biases in the types of genes most commonly isolated from various species and entered in the database. In many ways this would actually be a more interesting eventuality, and we are actively investigating this possibility. It should also be emphasized that the training and test results depend upon the integrity of the database annotations of coding sequence, which is certainly in doubt in many cases; indeed, it must be said that GenLang is not trained to recognize authentic genes but rather GenBank feature tables.

It is apparent that a general-purpose parser such as GenLang can, with appropriately-designed grammars, produce results comparable to special-purpose gene prediction algorithms. While the latter are specifically designed and optimized to exhaustively search the combinatorial space of exon assemblies, the parser uses a grammar formalism that serves equally well for other higher-order structures such as tRNA genes, and moreover examines only the first 100 complete parses produced. Compare this with GeneID, which typically evaluates tens of thousands of gene models each of which includes multiple "equivalent" exon clusters [9], or with GeneParser, which uses a dynamic programming matrix to literally examine every possible assignment of intron and exon boundaries in a sequence [21]. The smaller number of parses examined by GenLang is in fact necessitated by the well-known time penalty paid by general-purpose parsers as compared to fully customized code, though in fact GenLang, which averaged 2.5–3 minutes total per sequence on a SPARCstation 10, is less than an order of magnitude slower than the version of GRAIL we tested on the same platform. (More recently, we have ported the system to a native code compiler and achieved a two-fold speedup, while collaborators have developed parallel versions that promise even better performance.)

Performance comparisons are problematic in this domain, for reasons enumerated above, and also because these programs were designed for slightly different problems; for example, GeneID was meant to take pre-mRNA as input, and GeneParser was intended only to recognize internal exons, though both have been generalized to deal with genes embedded in intergenic sequences as well. It is possible that these other programs may prove superior on sequences longer than 20,000 bases (the limit imposed in the present study), and in particular on genes with very large numbers of exons, since GenLang depends upon backtracking left-to-right search. However, we feel that by taking advantage of the chart parser and designing grammar rules specialized for scanning lengthy sequences [19], we will be able to adapt GenLang to deal with such complex genes effectively, though perhaps not at the level of the intrinsically bottom-up approaches of other programs.

GenLang could return multiple genes in a sequence with some simple extensions, and in fact we have demonstrated this previously with multiple parses in a globin gene region [20]. Partial genes are more problematic, since the grammar represents a model of an entire gene, but grammars are easily written which span only subtrees. Pseudogenes or disease gene alleles with mutations that affect translation would also not be recognized (at least in their entirety) by the grammar described here, but again we have shown that similar grammars can be "relaxed" to allow for the detection of untranslateable messages, such as are produced with certain splicing defects [20]; all that is required is that the forms of mutations be modelled with the grammar as well, by methods we have described elsewhere [16, 18].

The inherent non-determinism of the GenLang parser may prove to be an advantage in addressing alternative splicing. In several alternatively-spliced genes we have examined, the known gene products all ranked high in the parse ordering, particularly when the mode of alternative splicing fit the backtracking scheme well (as in the three alternative final exons of drosophila aldolase). One might even argue that the left-to-right parsing paradigm in some way represents a better model of processive aspects of gene expression, known and perhaps unknown. There may be evidence of this in the fact that, of the up to 100 parses returned for each sequence, the minimal cost parse was in fact found relatively early in general: on the 26th parse for the vertebrate species on average, and on only the 12th parse for the other groups. For drosophila in particular, 42% of entries in the test sets produced the authentic gene as the *first* parse returned, though for only 31% was this also the minimum cost parse. (For all species, the *best* parses returned were significantly better than the *minimum cost* parses, e.g. for human data the best parses would have produced a correlation coefficient of 0.85 and an exon fraction of 0.59, indicating that there is room for improvement in the evaluation of gene structures generated by the grammar.)

In any case, we feel that the grammar representation constitutes an excellent foundation for further work in this domain. Not only are grammar rules intrinsically modular, hierarchical, and well-suited for rapid prototyping, but they have proven to be a suitable framework for embedding other algorithms as sensors, and for managing the combination of evidence from them. We note that the sensors used thus far are not among the most sophisticated currently under study, which include compositional measures drawn from signal processing and information theory (reviewed in [4]) and signal measures based on connectionist and classification techniques from machine learning (e.g. [1, 11]). The ability to apply such advanced sensors in a "plug and test" mode in a variety of grammar architectures and evidence combination schemes should allow them to be used to the best effect. Moreover, we believe we have yet to take full advantage of the capacity of grammars to represent the syntactic complexity and diversity that may be expected in this domain. As information about adjacent regulatory regions accumulates, and as models of splicing become more elaborate, the flexibility of grammars should increasingly come to the fore in representing and predicting gene structure.

The GenLang parser used for this work is available in the form of Quintus Prolog source code, and the system has recently been ported to the less expensive SICStus environment as well. Grammars described are also freely available, and runtime versions are currently under development. Contact D.B.S. at the address above or by electronic mail at dsearls@cbil.humgen.upenn.edu.

Acknowledgements

The authors thank Kyle Hart, Kevin Atteson, Jim Fickett, Chris Overton, and Charles Bailey for helpful discussions and other contributions. This work was supported by grant number DE-FG02-92ER61371 from the U.S. Department of Energy.

References

[1] S. Brunak, J. Engelbrecht, and S. Knudsen. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *J. Mol. Biol.*, 220:49–65, 1991.

[2] J.-M. Claverie, I. Sauvaget, and L. Bougueleret. k-Tuple frequency analysis: From intron/exon discrimination to T-cell epitope mapping. *Methods in Enzymology*, 183:237–252, 1990.

[3] J. W. Fickett. Recognition of protein coding regions in DNA sequences. *Nucleic Acids Res.*, 10:5303–5318, 1982.

[4] J. W. Fickett and C.-S. Tung. Assessment of protein coding measures. *Nucleic Acids Res.*, 20(24):6441–6450, 1992.

[5] C. A. Fields and C. A. Soderlund. gm: a practical tool for automating DNA sequence analysis. *CABIOS*, 6(3):263–270, 1990.

[6] K. S. Fu. *Syntactic Pattern Recognition and Applications.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

[7] M. S. Gelfand. Computer prediction of the exon-intron structure of mammalian pre-mRNAs. *Nucleic Acids Res.*, 18:5865–5869, 1990.

[8] M. S. Gelfand and M. A. Roytberg. A dynamic programming approach for predicting the exon-intron structure. *BioSystems.*, 30:173–182, 1993.

[9] R. Guigo, S. Knudsen, N. Drake, and T. Smith. Prediction of gene structure. *J. Mol. Biol.*, 226:141–157, 1992.

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

[11] M. Kudo, S. Kitamura-Abe, M. Shimbo, and Y. Lida. Analysis of context of 5' splice site sequences in mammalian mRNA precursors by subclass method. *CABIOS*, 8(4):367–376, 1992.

[12] R. H. Lathrop, T. A. Webster, R. Smith, P. Winston, and T. F. Smith. Integrating AI with sequence analysis. In L. Hunter, editor, *Artificial Intelligence and Molecular Biology*, chapter 6, pages 210–258. AAAI Press, 1993.

[13] B. W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta*, 405:443–451, 1975.

[14] F.C.N. Pereira and D.H.D. Warren. Definite clause grammars for language analysis. *Artif. Intell.*, 13:231–278, 1980.

[15] 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.

[16] 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.

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

[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] E. E. Snyder and G. D. Stormo. Identification of coding regions in genomic DNA sequences: an application of dynamic programming and neural networks. *Nucleic Acids Res.*, 21:607–613, 1993.

[22] R. Staden. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Res.*, 12:505–519, 1984.

[23] G. D. Stormo. Consensus patterns in DNA. *Methods Enzymol.*, 183:211–221, 1990.

[24] E. C. Uberbacher and R. J. Mural. Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. *Proc. Nat. Acad. Sci. USA*, 88:11261–11265, 1991.