Knowledge Discovery in GenBank

Jeffrey S. Aaronson, Juergen Haas & G. Christian Overton
Department of Genetics
University of Pennsylvania School of Medicine
Room 475, Clinical Research Building
422 Curie Boulevard
Philadelphia, PA 19104-6145
Internet: coverton@cbil.humgen.upenn.edu

Abstract

We describe various methods designed to discover knowledge in the GenBank nucleic acid sequence database. Using a grammatical model of gene structure, we create a parse tree of a gene using features listed in the FEATURE TABLE. The parse tree infers features that are not explicitly listed, but which follow from the listed features. This method discovers 30% more introns and 40% more exons when applied to a globin gene subset of GenBank. Parse tree construction also entails resolving ambiguity and inconsistency within a FEATURE TABLE. We transform the parse tree into an *augmented* FEATURE TABLE that represents inferred gene structure explicitly and unambiguously, thereby greatly improving the utility of the FEATURE TABLE to researchers. We then describe various analogical reasoning techniques designed to exploit the homologous nature of genes. We build a classification hierarchy that reflects the evolutionary relationship between genes. Descriptive grammars of gene classes are then induced from the instance grammars of genes. Case based reasoning techniques use these abstract gene class descriptions to predict the presence and location of regulatory features not listed in the FEATURE TABLE. A cross-validation test shows a success rate of 87% on a globin gene subset of GenBank.

1 Introduction

GenBank, the primary worldwide repository for nucleic acid sequence data, contains information on virtually all nucleic acid sequence that has been determined [Burks et al., 1991]. Each GenBank entry contains a FEATURE TABLE, a list of biologically significant features that, taken together, constitute GenBank's description of the structure of the sequence. Unfortunately, many GenBank entries suffer from incomplete, noisy (ambiguous or contradictory), and erroneous listings in the FEATURE TABLE. To a degree, these types of errors are inevitable in any large and complex database. The problem is exacerbated by the necessity of incorporating direct submissions from investigators into the database, without which GenBank would fall hopelessly behind in its effort to keep pace with the growing rate of sequence determination. However, investigators are often unfamiliar with the GenBank data description language, and as a result, fail to clearly represent their data. Typical obfuscations include missing features (e.g. introns not listed), mislabeled features (e.g. mRNA instead of prim_transcript or exon), incompatible boundary specifications among multiple features (e.g. between exon and CDS), and relegation of significant information (e.g. gene names in multigene entries) to free-text in the comment fields. Fortunately, much of this lost information can be recovered through analysis of the explicit information in the FEATURE TABLE to infer the implicit information. If we are to realize the full potential of GenBank, it is imperative that we develop tools that can discover this implicit data within GenBank.

We have developed several software tools in our laboratory designed to support this pursuit, chief among these is QGB [Overton et al., 1993], a system for performing complex queries on the information stored in flat-file and relational database versions of GenBank. Using a logic grammar as a model of gene structure, QGB corrects and disambiguates the listed features, discovers latent knowledge implicit in the FEATURE TABLE, and produces an idealized, augmented FEATURE TABLE as output. Queries in QGB, formulated in an SQL-like syntax, can be directed against the hierarchical sequence structures deduced by the logic grammar parser as well as other information in GenBank. A QGB query representing "return the locus ID, the definition line, and 10 bp 5' and 20 bp 3' to the 5'splice junction for all splice sites in all nonmammalian genes with complete coding sequences" would be constituted as

```
SELECT locus.id, definition,
  5'splice_site = JUNC(-10,pt:exon,pt:intron,20)
FROM '/databases/gbrel74/*.seq',
TO    myresults,
WHERE organism =\= mammalia AND
      definition AMONG
  ("complete cds" OR "complete coding sequence").
```

One of our major long-term goals is to apply computational approaches to the analysis and understanding of eukaryotic gene regulation. As part of this effort, we are taking various approaches towards discovering transcription elements and other regulatory signals in uncharacterized and partially characterized DNA sequences. Prediction of regulatory signals is of enormous practical value to researchers who can use this information to focus their costly and time-consuming experimental efforts on restricted regions of the DNA. We illustrate how our system can be used to automate the task of pattern recognition in the case where there are too few well-characterized examples of the regulatory sequences to apply statistical based machine learning methods towards inducing a pattern descriptor (see [Dietterich, 1990] for an overview of the requirements for statistical machine learning methods). As previously described [Overton & Pastor, 1991, Pastor et al., 1991], we have turned to a variant of **Case Based Reasoning** (CBR) [Kolodner, 1985, Kolodner et al., 1985], a form of reasoning by analogy, in this situation. One advantage of CBR is that it can succeed with only a few well-characterized examples if the uncharacterized test cases are sufficiently similar to some members of the example set. To do this, a CBR system makes use of domain knowledge, i.e., the similarity of the test case to some member of the case database, to replace the need for an accurate general gene structure model with an accurate local model. Methods of reasoning by analogy work well in this domain because similar biological systems are often **homologous**, that is derived from a common evolutionary ancestor, rather than being merely analogous. Furthermore, the CBR methodology matches the line of reasoning often used by biologists in practice: find a well-understood system similar to the new system, hypothesize the existence of features in the new system based on the features of the known system, then design and perform experiments to test for those features in the new system.

In CBR, well-characterized "cases" are organized and indexed in a case database. On the basis of the index, database cases are found which are similar to a test case and then these similar cases are used as templates to reason about the properties of the test case. The indexing scheme in our system is based on a static classification hierarchy constructed for attributes representing protein similarity and species similarity. The hierarchy is equivalent to a case database and the process of classifying a test case in the hierarchy amounts to the step of finding the most similar cases.

The paper is organized as follows: We provide background and motivate QGB's grammatical representation of gene structure, and describe its application to GenBank FEATURE TABLEs in Section 2. Section 3 details the construction of a classification hierarchy of genes that reflects the homologous relationship between similar biological systems. In Section 4, we discuss how gene class descriptions are recursively induced in the hierarchy from the instance grammars of genes. Section 5 explains how one CBR technique utilizes descriptive grammars and another utilizes sequence similarity in order to predict unknown regulatory regions in genes, and Section 6 suggests an application of CBR for correcting existing feature descriptions, rather than predicting new gene features. Finally, results of applying the system to the globin gene family subset of GenBank are given in Section 7, and a discussion can be found in Section 8.

Gene Primer

Our current work has focused on the globin gene family, whose proteins' function to transport oxygen, and include the myoglobins, hemoglobins and leghemoglobins. Figure 1 shows an abstract view of the structure of a canonical β-hemoglobin gene, but the essential features of this gene are typical of the genes of higher organisms. Substrings of a gene contain two types of information: information specifying the sequence of the gene's protein product, which is contained within the exon subsequences of the primary transcript, and information needed to **regulate** the process of gene expression.

Several hundred classes of elements are known that are part of the apparatus that controls gene expression, and more are discovered each year. These transcription elements typically range in length from 4 to 20 nucleotides, a size consistent with their presumed role as sequence specific recognition sites for binding of regulatory proteins. Upstream (located on the 5'flank of the primary transcript) **promoter** signal regions and downstream (located on the 3'flank of the primary transcript) **terminator** signal regions respectively act to define the start and stop points of the primary transcript. The class of promoters includes subsequences such as the **TATA, CAAT** and **CACA** boxes, which are found across a wide range of genes and species, as well as rarer subsequences that restrict gene expression to a specific tissue in an organism. Proper expression of a gene is critically dependent on the organization of the promoter sequences.

2 Gene Structure as a Grammar

QGB restructures the FEATURE TABLE of each GenBank entry into a collection of relational tuples, which are then processed by the **Sequence Structure Parser** (SSP) component of QGB. The SSP attempts to construct a parse tree expressing the structure of the gene described in the GenBank entry, or several parse trees in case the GenBank entry describes a gene cluster (see [Searls, 1993] for a full discussion of grammatical representation of gene structure).

In order to accommodate the peculiarities of the "language" of this domain we developed a generalization of the DCG formalism [Pereira & Warren, 1980] as an implementation technique of SSPs. Standard

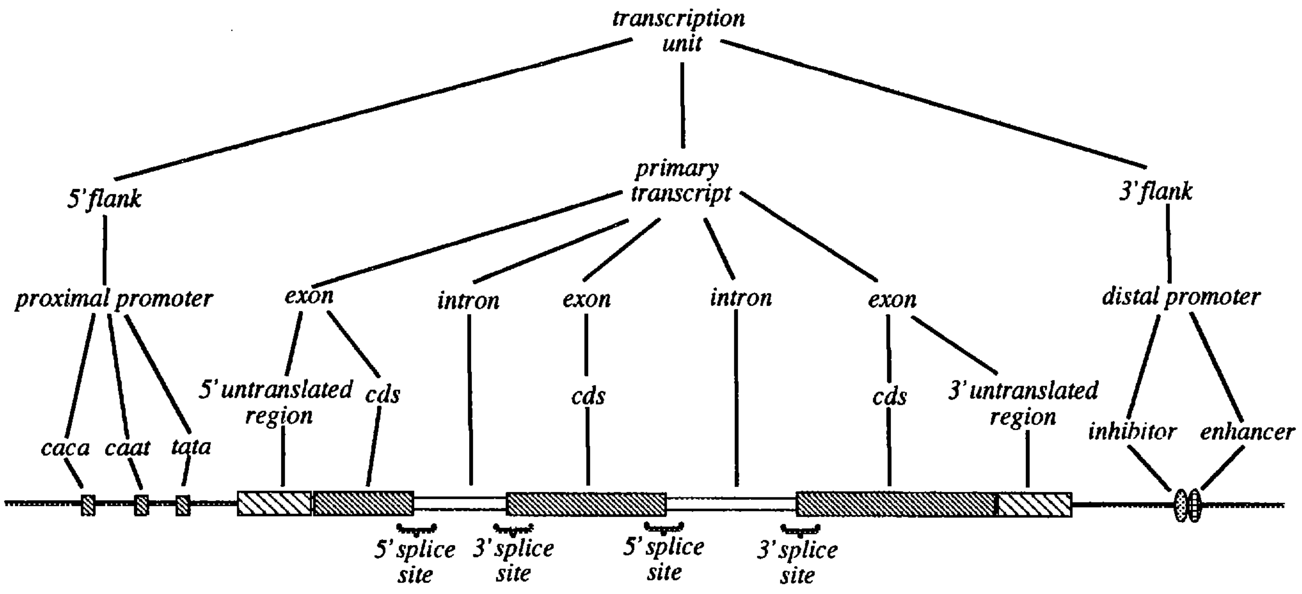

Figure 1: Idealized view of a parse tree for a typical eukaryotic protein coding gene.

DCG rules are of the form LHS => RHS, where the LHS (left-hand side) is a non-terminal and the RHS (right-hand side) any combination of terminals and non-terminals. **Terminals** correspond to words of the sentence being parsed (the leaf nodes of the parse tree), and **non-terminals** represent sets of phrases (subsequences of sentences) as defined by the grammar. Each interior node of a parse tree corresponds to a non-terminal as the sequence of terminals underneath such a node is one of the phrases of that non-terminal. The LHS non-terminal in the toplevel grammar rule is termed the start symbol.

In the context of nucleic acid sequences (NA) the distinction between terminals and non-terminals is less clear since genes can be described and investigated at various levels of abstraction. As shown in the parse tree of Figure 1, subsequences which may be considered as terminals in one context may become non-terminals in another (e.g., exons as subsequences of a primary transcript may be considered terminals, whereas exons would be non-terminals when parsed into coding sequences and untranslated regions).

Each feature of the FEATURE TABLE essentially describes one node (terminal or non-terminal) of the gene parse tree along with the DNA subsequence covered by it. These descriptions are often redundant, incomplete and even inconsistent, and the task of the SSP is to assemble complete parse trees expressing the same information in a structured non-redundant consistent fashion. Below is a simple example of an NA grammar rule expressing the fact that a transcription unit consists of a 5' flanking region, followed by a primary transcript and a 3' flanking region:

```
transcription_unit =>
   5'flank, primary_transcript, 3'flank.
```

Since grammatical elements (terminals and non-terminals) correspond to intervals of NA (sub)sequences [Overton et al., 1989], grammar rules can be naturally interpreted as interval relationships where '=>' means the interval on the LHS contains the intervals on the RHS ("part-whole" relationship), and a ',' between intervals means that the end of the first interval is the beginning of the second interval ("order of parts" relationship). Techniques have been developed for reasoning about temporal intervals [Allen, 1983], and these techniques can be extended to cover NA intervals. Incorporating these techniques into grammar rule formalisms makes it possible to model other interval relationships such as overlaps, starts, and ends [Pastor et al., 1991].

Contrary to standard parsers that take as input a list of terminals, the input to the SSP may contain non-terminals as well. To facilitate efficient processing the grammatical elements (features) on the input list are ordered by their start positions, lengths and ranks in the grammar hierarchy; for example, an exon occurs before a CDS fragment with the same boundaries. The square bracket notation, [], is used to remove and add elements to the input list. When used on the RHS of a rule, they remove grammatical elements, and when used on the LHS they add elements. Therefore, an element can be removed, examined and replaced on the input list as in the following example which tests if the 5'flank boundary has been reached:

```
5'flank, [primary_transcript(S,E,I)] =>
   gap, [primary_transcript(S,E,I)].
```

A)
```
cluster(pos(<,[0,0]),pos(>,[1138,1138]),(
 t_u(pos(<,[0,0]),pos(>,[1138,1138]),(
  f_f(pos(<,[0,0]),pos(=,[97,97]),(
   gap(pos(<,[0,0]),pos(=,[26,26]),()
   promoter(pos(=,[26,26]),pos(=,[31,31]),()
   gap(pos(=,[31,31]),pos(=,[69,69]),()
   promoter(pos(=,[69,69]),pos(=,[72,72]),()
   gap(pos(=,[72,72]),pos(=,[97,97]),())
  p_t(pos(=,[97,97]),pos(=,[929,929]),(
   exon(pos(=,[97,97]),pos(=,[230,230]),(
    five_utr(pos(=,[97,97]),pos(=,[134,134]),(
     gap(pos(=,[97,97]),pos(=,[134,134]),())
    cds(pos(=,[134,134]),pos(=,[230,230]),())
   intron(pos(=,[230,230]),pos(=,[347,347]),(
    gap(pos(=,[230,230]),pos(=,[347,347]),())
   exon(pos(=,[347,347]),pos(=,[551,551]),(
    cds(pos(=,[347,347]),pos(=,[551,551]),())
   intron(pos(=,[551,551]),pos(=,[691,691]),(
    gap(pos(=,[551,551]),pos(=,[691,691]),())
   exon(pos(=,[691,691]),pos(=,[929,929]),(
    cds(pos(=,[691,691]),pos(=,[820,820]),()
    three_utr(pos(=,[820,820]),pos(=,[929,929]),(
     gap(pos(=,[820,820]),pos(=,[908,908]),()
     pA_sig(pos(=,[908,908]),pos(=,[914,914]),()
     gap(pos(=,[914,914]),pos(=,[929,929]),())))
  t_f(pos(=,[929,929]),pos(>,[1138,1138]),(
   gap(pos(=,[929,929]),pos(>,[1138,1138]),())))
```

B)
```
FEATURES             Location/Qualifiers
     CDS             join(135..230,348..551,692..820)
                     /codon_start=1
                     /translation= NOT SHOWN
     precursor_RNA   98..929
                     /note="primary transcript"
     mRNA            98..230
     mRNA            348..551
     exon            692..929
                     /number=3
```

C)
```
FEATURES             Location/Qualifiers
     trans_unit      <1..>1138
                     /gene="alpha-globin"
     5'flank         <1..97
     promoter        27..31
                     /sequence="ccaat"
     promoter        70..72
                     /sequence="ata"
     prim_transcript 98..929
                     /note="primary transcript"
     exon            98..230
     5'UTR           98..134
     CDS             join(135..230,348..551,692..820)
                     /codon_start=1
                     /translation= NOT SHOWN
     intron          231..347
     exon            348..551
     intron          552..691
     exon            692..929
                     /number=3
     3'UTR           821..929
     pA_sig          909..914
                     /sequence="aataaa"
     3'flank         930..>1138
```

Figure 2: The parse tree, the original FEATURE TABLE, and the augmented FEATURE TABLE for the HUMAGL1 α-hemoglobin gene. A) standard representation of the parse tree for HUMAGL1 generated by the SSP; B) the FEATURE TABLE as found in GenBank; C) a representation of the parse tree for HUMAGL1 as a corrected, augmented FEATURE TABLE. Note that the start position for each interval in the parse tree is one less than the start position in the corresponding FEATURE TABLE entry because the SSP indexes on the space between characters rather than the character itself.

where the logic variables S and E represent the start and end positions of the interval, and I provides context information about the features.

Alternative rule applications (disjunction) can be expressed as follows:

```
5'flank => promoter, 5'flank.
5'flank => gap, promoter, 5'flank.
```

and recursion is illustrated in this example:

```
primary_transcript =>
    exon, intron, primary_transcript.
primary_transcript => exon.
```

Practical grammars also need escapes to the underlying implementation language. Such escapes are also available in NA grammars to handle exceptional situations such as erroneous and missing input data.

We developed an NA grammar for the class of genes that code eukaryotic proteins. It has been successfully applied to a large number of eukaryotic globin genes such as the human α-hemoglobin gene entry (HUMAGL1) shown in Figure 2B. The parse tree generated from this table is shown in Figure 2A. Note that this parse tree includes two promoters and one polyA-signal that were inferred by the techniques discussed below. In addition, the SSP inferred that the label `precursor RNA` should be changed to `primary transcript`, two features listed as `mRNA` are actually exons, and two introns were missing from the original FEATURE TABLE.

This example illustrates only a few of the problems that complicate the development of grammars with broad coverage. Apart from mislabeling and omission of features, the SSP has to deal with FEATURE TABLEs describing multiple genes or alternative versions of genes (e.g., alternative splice sites, transcription start sites, and polyA additions sites). In such cases it is often necessary to refer to the qualifier fields to determine which gene a particular feature is part of. However, qualifier fields are used inconsistently by authors thus complicating the task of the SSP. We intend to incorporate a more powerful text understanding component to extract more of the available information.

3 Hierarchy Generation

Reasoning by analogy is a powerful technique when applied to the study of biological systems, due to the homologous relationship between similar biological systems. The homology is rooted in evolution, and thus is a particularly strong form of analogy. In order to take advantage of analogical reasoning methods, we must develop a representation of GenBank that directly reflects the homological relationships between entries. This section describes just such a tree-like classification hierarchy of the family of globin genes, which serves as the platform for several analogical reasoning methods described in later sections.

The classification hierarchy is a generalization-specialization hierarchy in which subsumption is strictly enforced and therefore models of gene structure housed at a gene class are applicable to specializations and instances of that class. This is a crucial point: enforcing subsumption guarantees that the structure of our knowledge base mirrors the organization of the biological knowledge that we are modeling. Two classification attributes were used: biotaxonomic class and protein class. The former is specified by the taxonomic path, genus and species listed in GenBank entries, and the latter is determined from a cluster analysis of a protein similarity measure. We use a hierarchical clustering algorithm, described fully in [Rajasekhar & Overton, 1993], related to those used by the PIR protein sequence database [Barker et al., 1987] and described by [Harris et al., 1992]. The domain theory expressed by this hierarchy is that genes from more closely related organisms and whose proteins are similar will have similar regulatory proteins and DNA sequences.

We used QGB to identify all the globin entries within GenBank release 74. We then used the corresponding protein entries in the GenPept FASTA file, version 74, produced by NCBI to generate pairwise alignment scores, using the FASTA program, **align0**, for all the globin products. The scores were then clustered, forming an initial classification based solely on protein similarity. The elements in each resulting leaf cluster were then hierarchically classified with respect to biotaxonomic relatedness. Thus, protein similarity dominates the coarse structure of the hierarchy, while biotaxonomic relatedness dominates the fine structure.

An interior node of the final tree represents a subclass of the globin genes, defined by the leaf nodes it scopes. The leaves correspond to the globin gene products that were input to the cluster analysis. Each leaf is annotated with the data from the associated GenBank entry, provided by the FFP, as well as the parse tree constructed by the SSP. Grammar rules specifying the instance grammar defined by the resident parse tree are also attached to each leaf.

We will see in Section 4 how grammars are merged up the hierarchy, providing a description at each interior node of the gene structure of the class of genes described by that node. Section 5 show how these grammar are utilized during analogical reasoning.

4 Grammar Induction

Analogical reasoning predominantly involves extrapolating known attributes of a given case to fill in the specification of an incomplete case. However, GenBank FEATURE TABLEs are vastly underspecified, providing only partial information for a typical gene instance. Thus, in general, gene instances are not sufficiently informative individually to serve as cases. If we are to make effective use of our hierarchy, we must construct more complete descriptions of gene instances to use as cases. Accordingly, we treat abstractions over multiple gene instances as our cases.

We construct these abstractions by beginning at the instance grammars at the leaf nodes and recursively inducing grammar rules at each interior node up the hierarchy, forming a descriptive theory at each node of the class of genes scoped by the node. A consensus sequence is formed for each feature during each induction step, and is attached to the induced grammar. Section 5 explains how this annotated generalized gene class description may then be used to fill in missing features of all of the genes scoped by that gene class.

It is important to note that we may wish to *disjoin*, rather than *merge* grammar rules of a certain feature when we reach a certain height in the classification hierarchy, as the gene class may contain sufficiently disparate structures that we may not want to form an abstraction [Overton & Pastor, 1991].

The general problem of inducing grammars from a set of legal training examples is exceedingly difficult. One of the foremost obstacles is that the space of possible grammars facing an unbiased inducer is usually quite large and for some grammar representations, such as context-free grammars, is infinite. Various possible solutions have been proposed (for example, see [Langley, 1987]) to counter this obstacle. However, we are dealing with a highly constrained problem [Haas et al., 1993], and so far have been able to design tractable induction algorithms for the grammar rules (e.g. 5'flank, 3'UTR) incorporated to this point.

5 Prediction of Regulatory Signals

Grammar Based CBR

The generalized abstract grammars discussed in Section 4 may be used to hypothesize features of genes that are underneath the corresponding node in the hierarchy. These hypotheses specify both the identity and the location of proposed features. Where a feature's location and size are identical in all similar genes, it is possible to give absolute start and end positions; this is typically not the case, and locations can therefore be predicted only relative to other features at the same level of the grammar, with a projected absolute location and size based on normalized location and size. Each grammar nonterminal is annotated with "positional references" defining the relative location of the corresponding feature as well as "instance sequences" defining the possible nucleotide sequences of that feature.[1] More precisely, the positional references of a feature are the distances of that feature from the start and end of the sequence covered by the next higher-level grammar nonterminal. E.g., consider the grammar rule:

```
3'UTR --> gap,
          polyA_sig(at:{[60,27,aataaa]}),
          gap.
```

The polyA-signal in this rule is annotated with its distance from the start and end of the **3'UTR** region which would be in this case **60** and **27** respectively. The sequence of this polyA-signal is specified as **aataaa**. Nonterminals of rules of abstract grammars in general have several such triples, one for each gene instance from which the abstract rule was derived. Using this information, features of gene instances with incomplete grammars can be predicted by "matching" rules of the abstract grammar with the corresponding rules of the instance grammars as illustrated in the following example:

Rule of abstract grammar:
```
(1)  5'flank -->
       gap,
       promoter(at:{...,[_,79,ccaat],...}),
       gap,
       promoter(at:{...,[_,35,tataaa],...}),
       gap.
```
Corresponding rule of gene instance grammar:
```
(2)  5'flank -->
       gap,
       promoter(at:{[_,37,tataaa]}),
       gap.
```
Rule inferred by applying abstract grammar to instance grammar:
```
(2') 5'flank -->
       gap,
       promoter(at:{[_,78,ccaat]}),
       gap,
       promoter(at:{[_,37,tataaa]}),
       gap.
```

[1]In the current implementation instance sequences are maintained only for small features such as promoters and polyA-signals.

Since the start of the 5' flanking region is not precisely defined, only the reference to the *end* of the 5' flanking region can be given for the above cases (only one of the two references is needed to compute the absolute location of the feature). The abstract rule **(1)** may be used to predict features in the instance rule **(2)** by matching the right-hand side of **(1)** with the right-hand side of **(2)**. This type of matching takes into account the positional references as well as the instance sequences: the positional references of the abstract rule serve to restrict the potential relative position of the feature; the exact position is determined by searching the predicted area for a sequence matching the instance sequence associated with the positional references. A feature of the abstract rule will be merged with an overlapping feature of the instance rule, retaining the positional references of the instance rule. In the above example the promoter at relative position **35** in rule **(1)** overlaps with the promoter at relative position **37** of rule **(2)**; therefore these two promoters are merged retaining **37** as positional reference. This leaves the first promoter of **(1)**, including the two neighboring gaps, to be merged with the first gap of **(2)**. Based on the reference information **[_,79,ccaat]** of the first promoter of **(1)**, a corresponding promoter is predicted for the gene modeled by **(2)** at 79 nucleotides to the left of the end of the 5' flanking region. If such a promoter actually exists in that area, its exact location can be determined by searching the area for the sequence **ccaat**. In the above example the actual location turns out to be one position to the left of the predicted location. Replacing the first gap of **(2)** with the result of this merge yields the refined grammar rule **(2')**.

If the sequence of a predicted feature can actually be located near the predicted position, it would be a strong indication that the predicted feature has the biochemical properties attributed to that type of feature (such as promotion of gene expression). However, such properties can be reliably confirmed only by means of experimental analysis. Consequently, unconfirmed hypotheses may persist in the knowledge base for indeterminate periods of time, and may in turn contribute to further hypotheses. For this reason, because of the possibility of changes to the underlying databases (e.g., GenBank updates), and to support explanation of the derivation of hypotheses, some form of truth maintenance [Doyle, 1979] is necessary. Truth maintenance in a database the size of GenBank is somewhat problematic but we are taking pains to preserve sufficient information during theory formation (grammar induction) and hypothesis generation to support limited forms of truth maintenance and explanation.

In order to search for an instance sequence near the predicted location, certain parameters have to be set to precisely define what we mean by "*near* a location" and "*matching* of two sequences". In addition, since in general there are several matches, it is necessary to rank them by quality; i.e., a score must be computed for each match so that the "best" match can be determined. The following heuristic takes into account the degree of mismatch and displacement of the predicted feature and was used to compute the data given in section 7:[2]

$$Score = ((100 - Mismatch)/100)/2^{(Off/10)}$$

where *Mismatch* is the percentage of mismatch and *Off* is the distance of the sequence from the predicted position. That is, the maximum score is 1; this score is reduced by the percentage of mismatch, and goes down exponentially when shifted from the predicted position in such a way that the score is reduced to half for every 10 positions off the predicted location. Matches are discarded if their mismatch score or the distance from the predicted position becomes too large. For the test runs discussed below sequences had to match at least 80% and must not be off the predicted position by more that 10 positions. The above heuristic was chosen because it has the desired characteristics to the extent that we understand the biological basis of the type of CBR discussed here. Numerous other scoring methods appear equally plausible at this stage and we will refine this heuristic as our understanding of the underlying biological mechanisms improves.

The reliability of predictions resulting from grammar based CBR can be judged in basically two ways: (1) Verification: If the relative location of the predicted feature is close to that suggested by a reference and there is a high degree of similarity between the two sequences then it is likely that the predicted feature is "correct", i.e., it marks a feature with the same type of function as the reference feature, particularly if several references imply the same prediction. (2) Cross-validation: By omitting one feature from the knowledge base/hierarchy at a time and testing whether the procedure predicts the omitted feature, one can compute a measure of reliability as the ratio of the number of omitted features that were correctly recovered and the total number of predictions. Those features that were not recovered correspond to "false negatives". Unfortunatedly, it is difficult for this application area to give a reliable estimate of the number of "false positives" since our data are highly incomplete (most regulatory signals are not marked in GenBank). A rough estimate of the number of "false positives" can be given by comparing the total number of predicted features for each feature type with the total estimated number of such features, taking into account the percentage of false negatives (see discussion in Section 7).

Sequence based CBR

If there is a high degree of similarity between the sequences of two genes, and if a particular feature is specified only for one of them, one can hypothesize a corresponding feature in the other gene. The position of the predicted feature is determined by aligning the two sequences and marking the boundaries of the feature already known in this alignment.

[2]The optimal values of the parameters of this formula are currently being determined experimentally.

```
GORGM12GLB:
                    2010        2020
              ... cttgaccaat agcct ...
                        <caat>
HUMGAMGLOA:
                    2006        2016
              ... cttgaccaat agcct ...
```

Figure 3: Sequence based CBR.

For example Fig. 3 shows a section of the alignment of the sequences of the human γ-globin genes (GenBank ID HUMGAMGLOA) and the gorilla γ-globin gene (GenBank ID GORGM12GLB). They match almost perfectly over more than 11000 base pairs. The promoters are listed only in the GORGM12GLB entry as illustrated for the CAAT signal in Fig. 3. However, given the high degree of similarity between the two sequences, one can predict with confidence that the HUMGAMGLOA sequence has the same promoter at the corresponding position (2011-2015). In this approach we make use of the fact that the sequences have already been structured as a parse tree and that they are organized in a hierarchy that groups genes of closely related organisms together. This makes it easy to find sequences that match to a high degree so that features specified in one entry can be hypothesized with high reliability in the other entry. Having the sequences structured as a parse tree allows us to find matching regions with little or no search since it is already known where major features, such as coding sequences, are located in each sequence. Typically, the procedure would start the comparison at the beginnings of the coding sequences of each gene as specified in the parse tree and proceed in both directions, allowing for a certain degree of mismatch and insertions/deletions.

Similar to predictions made by grammar based CBR, predictions made by sequence based CBR can be assessed in two ways: (1) Verification: A high degree of similarity between two sequences indicates that a feature that has been predicted at the same relative location in one sequence as the corresponding feature F in the other sequence has the same function as F. (2) Cross-validation: Reliability can also be tested by omitting known features and testing whether the procedure "rediscovers" them.

6 Correction of Feature Descriptions

A prerequisite for correct feature prediction is correct input data. Unfortunately, many feature descriptions from the input data bases contain mistakes or inconsistencies. However, it turns out that the same techniques used for predicting features can also be used to correct many such mistakes by isolating data that conflict with results from CBR. As discussed in Section 2, QGB can detect and correct with some reliability certain inconsistencies within a GenBank entry. However, many inconsistencies can only be detected by making comparisons between several entries.

Grammar based CBR may predict a feature that is inconsistent with a feature already listed. A closer examination of this discrepancy may reveal a problem in the entry that should be corrected before it is used for further CBR. For example the entry ALFLEGHEMA lists several alternative polyA-signals, none of them is consistent with the polyA-signal predicted by CBR. Therefore, in order not to degrade the performance of the CBR procedure, these polyA-signal entries should be removed from the input until more reliable data are available.

Sequence based CBR is also a powerful technique for detecting and correcting inconsistent feature boundaries where applicable. For example, if the sequences of several genes can be aligned with more than 95% agreement, and all boundaries of a particular feature are marked at the same relative locations except for one, there is a high probability this feature boundary is incorrect and should be at the same relative location as the others.

7 Results

We examined how well QGB did in discovering features not explicitly listed in the FEATURE TABLE but rather inferred by the SSP. Our globin data set contained a total of 437 introns and 665 exons. QGB inferred an additional 130 introns (30% more) and 267 exons (40% more) not listed in the FEATURE TABLE. These features would not be recognized or reported by the other sequence extraction tools currently available.

The set of globin entries for which complete parses were obtained has been organized in a hierarchy as described in Section 3. This hierarchy has been used as a data set for grammar based CBR. Reliability of this procedure has been measured by iteratively removing one of the 134 promoter and polyA-signals and testing whether the deleted feature is rediscovered by the CBR procedure. The CBR procedure was able to correctly predict 116 (87%) such features, and "missed" 18 (13%) of them (false negatives).

It is difficult to estimate the number of "false positives", i.e., wrong predictions, in this case, since most promoter and polyA-signals are not annotated in GenBank. That is, if the system predicts a feature that is not listed in the GenBank entry one cannot tell whether that feature doesn't exist or whether it exists but has not been listed. However, if we assume that each gene on the average has at least three promoters and one polyA-signal, the number of false positives in this test run can be estimated in the following way: The grammar rules induced for the root node assumed three promoters and one polyA-signal. Therefore, for each gene the CBR procedure will try to predict as many promoters and polyA-signals as are necessary to obtain a total of three promoters and one polyA-signal. This means, on average, the number of predicted features will be less than the number of existing but unknown features. This implies that the number of false positives is less than the number of false negatives. Given our experimental result of 13% false negatives we can therefore estimate the number of false positives to be less than 13%.

Based on these results, we can now predict previously unknown features with a certain degree of confidence. Application of grammar based CBR to all entries in the hierarchy yielded a total of 312 new promoter and polyA-signals. According to the above analysis at least 271 (87%) of these features can be expected to be "real".

The procedure for sequence based CBR has been applied to a cluster of 111 globin genes from our data set. Twenty of these GenBank entries have information about promoters or polyA-signals. The total number of promoters and polyA-signals in this set is 63. During cross-validation sequence based CBR was able to recover 50 (79%) of these features. 13 (21%) were missed and no false predictions could be identified. Actual application of sequence based CBR to the cluster of 111 globin genes yielded 149 new features (127 promoters and 22 polyA-signals).

After inferring features through CBR the system augments the grammars accordingly. The grammars are then converted back into parse trees, and finally, the parse trees, supplemented by information from GenBank entry headers, are mapped into corrected, augmented versions of the FEATURE TABLE (Figure 2C), a clear improvement over the incomplete and inconsistent FEATURE TABLE of the original GenBank entry (Figure 2B). Only a few additions to the feature and qualifier types are required in the improved version. The augmented tables provide a consistent, uniform representation of features across all entries and further, can be used to directly regenerate the parse tree without recourse to the SSP thus significantly reducing query response times.

8 Discussion and Future Work

It is important to note that the techniques for inferring, discovering, and correcting genetic knowledge discussed in this paper complement and enhance each other in various ways. For example, sequence based CBR may use the grammars constructed by the sequence structure parser to correct feature boundaries and infer features that can be inferred with high confidence. Next, the grammar induction algorithm would infer grammars for all nodes in the hierarchy which will be used by grammar based CBR to infer additional features and make further corrections of feature boundaries. The GenBank reconstruction routine would now replace the original GenBank entries with improved versions. The improved entries will allow the parser to produce improved parse trees and grammars which may yield an improved hierarchy and improved CBR results. The process can be repeated until the hierarchy and its contents stabilizes.

The structure of the hierarchy itself may be influenced by the result of the sequence structure parser since, for example, mislabeled coding regions in a gene cluster may make it difficult to determine the complete coding sequence for a particular gene. Since the clustering algorithm is based on protein sequence similarity, an incorrect parse of the coding sequences may lead to incorrect classification of that entry in the hierarchy which may prevent useful CBR inferences.

The predictive power of the hierarchy is correlated strongly with its structure. We are automating techniques designed to eliminate noise and redundancy from our data set, thereby improving the integrity of our classification hierarchy. We are exploring including additional attributes, such as time and tissue of development, into our classification hierarchy.

Finally, we intend to set up the system in such a way that it automatically changes various parameters of the classification algorithm and CBR procedures and evaluates the performance of the modified system so that the optimal parameter settings can be determined.

Acknowledgements This work was supported by a grant (NIH RO1 RR04026) from the National Center for Research Resources, National Institutes of Health.

References

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

W. C. Barker et al. Protein Sequence Database of the Protein Identification Resource (PIR). CODATA Exchange Format Specification Document CXFSD-0687, 1987

Burks, C., Cassidy, M., Cinkosky, M. J., Cumella, K. E., Gilna, P., Hayden, J. E-D., Keen, G. M., Kelley, T. A., Kelly, M., Kristofferson, D., and Ryals, J. (1991) GenBank. NAR 19 (Suppl):2221-2225.

Cinkosky, M. J., Fickett, J. W., Gilna, P., and Burks, C. (1991) Electronic Data Publishing and GenBank Science 252, 1273-1277.

Thomas G. Dietterich. (1990) *Machine Learning*, vol. 4, pp. 255-306. Annual Reviews Inc., 4139 El Camino Way, P.O. Box 10139, Palo Alto CA 94303-0897.

J. Doyle. (1979) A Truth Maintenance System. *Artificial Intelligence*, 12(3).

Haas, J., Aaronson, J. S. and Overton, G. C.(1993). Grammar Induction on Gene Structure Grammars. Technical report, University of Pennsylvania. In preparation.

Harris, N., Hunter, L., States, D. (1992). Megaclassification: Discovering Motifs in Massive Datastreams. In Tenth National Conference on Artificial Intelligence (AAAI), pp. 837-842. San Jose, CA: AAAI Press.

Janet L. Kolodner. (1985). Experiential Processes in Natural Problem Solving. Technical Report GIT-ICS-85/23, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA 30332.

Janet L. Kolodner, Robert L. Simpson, Jr., and Katia Sycara-Cyranski. (1985) A Process Model of Case-Based Reasoning in Problem Solving. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 284–290. Morgan Kaufmann.

Pat Langley. (1987). Machine Learning and Grammar Induction. *Machine Learning*, 2:5–8.

Overton, G. C., Koile, K., and Pastor, J. A. (1989). GeneSys: A knowledge management system for molecular biology. In Bell, G. and Marr, T., editors, *Computers and DNA*, pages 213–240, Reading, MA. Addison-Wesley.

Overton, G. C. and Pastor, J. A. (1991). A Platform for Applying Multiple Machine Learning Strategies to the Task of Understanding Gene Structure. In *Proceedings of the 7th Conference on Artificial Intelligence Applications*, vol. I:Technical Papers, pp 450-457.

Overton, G. C., Aaronson, J., Haas, J., and Adams, J. (1993). QGB: A System for Querying Sequence Database Fields and Features. Submitted for publication.

Pastor, J. A., Koile, K., and Overton, G. C. (1991). Using analogy to predict functional regions on genes. In *Proceedings of the 24th Hawaii International Conference on System Science*, volume I, pages 615–625.

Pereira, F. C. N. and Warren, D. H. D. (1980). Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Transition Networks," *Artificial Intelligence*, vol. 13, pp. 231-278.

Rajasekhar, R. and Overton, G. C. (1993). An Unsupervised Hierarchical Clustering Algorithm for Unequal Length Vectors. Technical report, University of Pennsylvania. In preparation.

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