A COMPUTATIONAL SYSTEM FOR IDENTIFYING CIS-REGULATORY ELEMENTS IN CLOSELY RELATED GENOMES

By

Thomas Michael Smith

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Computer Science

Approved by the
Examining Committee:

_____
Lee A. Newberg, Thesis Adviser

_____
Mukkai S. Krishnamoorthy, Thesis Adviser

_____
Kristin P. Bennett, Member

_____
Joyce H. Diwan, Member

_____
David L. Spooner, Member

Rensselaer Polytechnic Institute
Troy, New York

April 2006
(For Graduation May 2006)

Contents

List of Tables

List of Figures

Acknowledgments

The author would like to thank Professors Newberg and Krishnamoorthy for valuable assistance with both coursework and research over the past several years. The author would also like to thank Prof. William Thompson of Brown University, Dr. Lee Ann McCue of Pacific Northwest National Laboratory, and Dr. Sean Conlan of Columbia University. The author would finally like to thank Prof. Charles "Chip" Lawrence (currently at Brown University) for supporting this work and helping to create the pioneering Computational Molecular Biology curriculum within the Computer Science Department at Rensselaer Polytechnic Institute.

The author additionally thanks the Computational Molecular Biology and Statistics Core of the New York State Department of Health Wadsworth Center for Laboratories and Research for the use of their computer cluster.

Abstract

Scientists are only beginning to develop methods for whole genome comparisons. Computational methods which do not incorporate evolutionary relationships in their models are inadequate when there is high correlation between the DNA sequence data from closely related species. New genomes from various species are often sequenced specifically because of their relationships to other species as researchers wish to learn how species are differentiated at the DNA sequence level.

Current methods do not provide a computational platform for efficiently analyzing data from many genomes simultaneously. Genomic scale analysis tends to result in hundreds of thousands of files across gigabytes of disk space with much redundancy. As the library of complete genomes grows, researchers will analyze larger sets of more complex genomes. A software system which scales to facilitate these multiple genome analyses will be necessary.

This thesis aims to provide a computational system for identifying *cis*-regulatory elements in closely related genomes. As a demonstration, an analysis of *cis*-regulatory elements in *E. coli* and related bacterial genomes is provided. Previous computational methods for such an analysis are somewhat ineffective because the species are closely related. This closeness causes statistical problems due to the high correlation between the sequence data in the genomes.

Methods for discovering potential *cis*-regulatory elements from DNA sequence data are reviewed, and a new Gibbs Sampling method is provided to address the correlation problem by accounting for evolutionary distances between species. A new software system is also shown to provide a computational platform necessary for more complex, collaborative analyses in the future.

The software framework provides an object-oriented representation for the Gibbs Phylogenetic Sampler and generically lays the groundwork to accommodate other statistical bioinformatics applications. Many sampling strategies, biological data structures, and mathematical models may be shared between various analysis applications.

On synthetic data, the new Gibbs Sampling method demonstrates greater positive predictive value and significantly greater specificity than other, comparable methods. On real data, the new method also produces encouraging results. A comparison of features between this and other methods is also provided.

Chapter 1

Introduction

1.1 Motivation

Researchers wish to study closely related genomes. For example, seven genomes from the Shewanella genus have recently been sequenced (Heidelberg *et al.*, 2002), and several more are currently being considered for sequencing. Shewanella bacteria have the unusual ability to reduce metal contaminants (Nealson & Little, 1997), so they may be useful for bioremediation (Lovley & Lloyd, 2000). In particular, the United States Department of Energy is interested in using such microbes to clean up contaminated sites. Since Shewanella are members of the gamma proteobacteria class, several slightly more evolutionarily distant genomes are already available for computational comparison; however, none of the other gamma proteobacteria can reduce metals, so very few inferences about the genetics behind this ability can be made.

Scientists are only beginning to develop methods for whole genome comparisons. Gene expression, which will be discussed in further detail in section 1.2.1, is an area of great interest for such comparisons. Computational methods which do not incorporate evolutionary relationships in their models are inadequate when there is high correlation between the DNA sequence data from the various species. This does not appear to be a significant problem for species separated at the class level taxonomically, but it has been shown to be a problem at the family level with at least one method (McCue *et al.*, 2002). Since Shewanella bacteria are all in the same genus (which is more specific than family), high correlation between their sequence data is expected. Current methods do not adequately account for this correlation.

Additionally, current methods do not provide a computational platform for efficiently analyzing data from many genomes simultaneously. Genomic scale analysis

tends to result in hundreds of thousands of files across gigabytes of disk space with much redundancy. With cluster computing, much of this redundant information is also transmitted over a local area network. Aside from the computational inefficiency of such a system, the process and final results are often difficult for humans to interpret or reproduce. As the library of complete genomes grows, researchers will analyze larger sets of more complex genomes. For example, the "Zoo Project" (Thomas *et al.*, 2003) is expected to promote the significantly larger analysis of mammalian genomes. Such projects require scaling from the analysis of millions of positions in DNA sequences to the analysis of billions. A software system which scales to facilitate these multiple genome analyses will be necessary.

1.2 Background

1.2.1 Gene Expression

A gene is a piece of a DNA molecule which encodes a segment of RNA or instructions for creating a protein. Gene expression is the process of creating proteins within a cell. The set of proteins within a cell is of fundamental importance to cell function and differentiation. Gene expression begins with the transcription of a gene to messenger RNA, continues through translation of messenger RNA to a polypeptide, and ends with the folding of one or more polypeptides into a protein.

Transcription is the process in which an RNA Polymerase protein synthesizes messenger RNA from template DNA. RNA Polymerase begins by binding to the DNA upstream of the gene to be transcribed. The region of DNA where RNA Polymerase binds is known as the promoter. RNA Polymerase moves along the template DNA from the promoter across the length of the gene, assembling a complementary messenger RNA molecule.

Transcription regulation is one of the most important mechanisms for determining if a gene will be expressed. Transcription factors are proteins which regulate the transcription of DNA to messenger RNA by binding to the DNA molecule, usually upstream of the gene to be transcribed. The region of DNA where transcription factors bind is known as the *intergenic* region because it is located between genes. The presence or absence of these transcription factors affects the ability of RNA Polymerase to bind to the promoter and initiate transcription. The classic example of regulation of the *lac* operon illustrates this (Jacob & Monod, 1961).

Please see *Genes* (Lewin, 2000), *Molecular Biology of the Cell* (Alberts *et al.*,

2002), or *Biochemistry (Voet & Voet, 2004)* for more details on gene expression.

1.2.2 *Cis*-Regulatory Elements

The DNA binding sites of transcription factors, known as *cis*-regulatory elements, are characterized by common, subtle patterns (or *motifs*) of nucleotides shared by all sites to which a given transcription factor may bind. The method by which a protein recognizes its binding site is not well understood, and this makes it difficult to predict which proteins could bind to a particular segment of DNA (Pabo & Sauer, 1984). It has been shown that there is no simple "recognition code" which maps the binding of amino acids in proteins to corresponding nucleotides in DNA (Matthews, 1988); however, it is possible to determine probabilistic recognition codes for high-affinity binding interactions by analyzing experimental binding data(Benos *et al.*, 2002).

High throughput experiments to test for transcription factor binding *in vitro* are difficult due to the combinatorial problem of selecting transcription factors, potential binding regions, and environmental binding conditions. Since there is no simple recognition code, and experimental binding data across an entire genome is difficult to collect, we focus on the DNA sequence alone. Specifically, we focus on sequences which may be regulated by the same transcription factor. If multiple sequences actually do bind the same factor, there is likely to be a common subsequence. This common subsequence will not be exactly the same in each sequence, so transcription factor binding sites are often probabilistically modeled as a product multinomial distribution, which may also be represented by a position weight matrix. This matrix contains the probability of each nucleotide at each position in the sequence of the binding site, with the assumption that the distributions at each position are statistically independent. It is also convenient to represent a motif visually as a *sequence logo* (Schneider & Stephens, 1990). The sequence logo also accounts for *information content* at each position, which will be discussed briefly in section 2.1.1.1. Please see figure 1.1 for an example sequence logo.

1.2.3 Phylogenetic Relationships

Evolutionary relationships between species are called phylogenetic relationships. Fundamentally, these relationships measure the time between speciation events where ancestral species split into distinct descendant species. These relationships are evident in the taxonomic classification system. Before nucleic acids could be sequenced, phylogenetic relationships were inferred by observing different quantitative and qual-

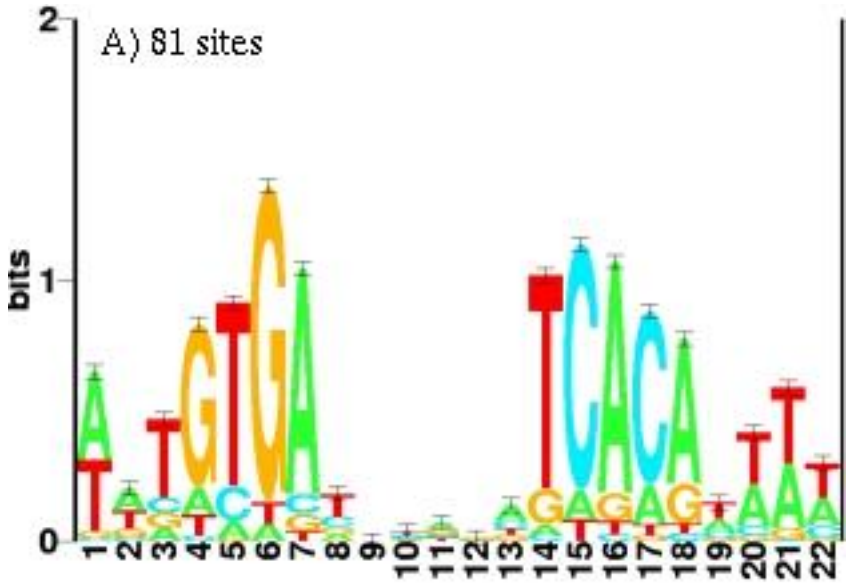


Figure 1.1: Sequence Logo for Cyclic AMP Receptor Protein (Smith, 2003)

itative characteristics between species and classifying them. For plants and animals, morphological characteristics could be used. For bacteria, a series of laboratory tests was established to aid in classification (Holt, 1994). The fossil record was also useful for inferring a time line for phylogenetic relationships between non-bacterial species.

Classification alone is not precise enough to develop a clear picture of evolution, so scientists began to focus on genetic sequences (Zuckerkandl & Pauling, 1965). Specifically, 16S ribosomal RNA (rRNA) genes became the molecules of choice for phylogenetic analysis (Woese *et al.*, 1975; Woese & Fox, 1977). 16S rRNA genes exist in all organisms, are long enough to contain a reasonable amount of historical information, and have a very slow rate of change over time. These properties make 16S rRNA genes ideal candidates for analysis to infer phylogenetic relationships (Fox *et al.*, 1980; Olsen *et al.*, 1994). An on-line database cataloging over 97,000 (updated since publication) bacterial 16S rRNA gene sequences has been created for the study of phylogeny (Cole *et al.*, 2003). Computational methods can be applied to these sequences to create trees such as the one in figure 3.1. Intuitively, but not precisely, the distances between the leaves of the tree represent time. Unrooted trees are often used because of ambiguity in the time line which will be discussed in section 2.2.3. Mathematically, the distances are proportional to the expected number of substitutions per 16S rRNA gene sequence position between the species under an assumed model.

Chapter 2

Historical Review

2.1 Algorithms for Detecting *Cis*-Regulatory Elements

Several approaches have been taken to develop *local multiple alignment* algorithms to discover both motifs and binding sites (*cis*-regulatory elements, instances of motifs) from sequence data. Given several sequences which are believed to contain the same *cis*-regulatory element, a local multiple alignment algorithm attempts to find similar DNA subsequences by shifting one or more windows in each sequence back and forth across the sequence and evaluating the similarity of the subsequences contained in the windows. Simple string search algorithms are ineffective due to variability at each nucleotide position in a *cis*-regulatory element, so mathematical optimization methods are necessary.

2.1.1 Early Optimization Algorithms

2.1.1.1 The CONSENSUS Algorithm

The first notable local multiple alignment algorithm was developed by Stormo and Hartzell in 1989 (Stormo & Hartzell, 1989). Called CONSENSUS, their algorithm maximizes the *information content* of potential *cis*-regulatory elements. In this case, information content is proportional to the logarithm (base 2) of the ratio of the observed nucleotide frequencies to their frequency throughout a whole genome. The algorithm begins by assuming that any subsequence of a given length in the first sequence may be a binding site. A position weight matrix is created for each site, and a value of 1 is assigned to nucleotides present at each position, with values of 0 for the

rest. Another sequence in examined, and all of its subsequences of the given length are compared to the matrices. Various heuristics to maximize information content are used to update the matrices by adding the nucleotides present in the new sequence. This process is repeated until all of the sequences have been used. The CONSENSUS algorithm has since been refined to allow gaps in motifs (Hertz & Stormo, 1994) and evaluate statistical significance of potential binding sites (Hertz & Stormo, 1999).

2.1.2 Expectation Maximization Algorithms

2.1.2.1 Lawrence & Reilly's EM Algorithm

Early statistical methods used expectation maximization (EM) techniques. Lawrence and Reilly's EM algorithm (Lawrence & Reilly, 1990) begins with a random product multinomial distribution for a motif. Given this distribution, the algorithm calculates the probability of a binding site beginning at each position in each sequence. These probabilities are used as weights for calculating the expected value for the nucleotide distributions in the motif. The algorithm selects binding sites which maximize the likelihood of the population distribution by iterating this process until the expected values stabilize.

2.1.2.2 The MEME Algorithm

In 1994, Bailey and Elkan developed an expectation maximization algorithm known as MEME (Bailey & Elkan, 1994). The MEME algorithm is similar to Lawrence and Reilly's EM algorithm, but it removes the constraint of requiring exactly one site per sequence. Instead of focusing on just the product multinomial motif, MEME uses a mixture distribution where it also estimates the most likely background as a single multinomial distribution and the probability that a position follows the background distribution or the motif distribution.

2.1.3 Gibbs Sampling Algorithms

2.1.3.1 The Gibbs Site Sampler

Newer methods use Gibbs Sampling algorithms, and they have been shown to be quite effective. In 1993, a Gibbs Sampling algorithm was developed to discover a single motif and its corresponding instances (sites) in a set of sequences (Lawrence *et al.*, 1993). Each sequence is assumed to have exactly one site. This algorithm is known as the Gibbs Site Sampler. The probabilities are calculated almost identically

as in the Lawrence and Reilly EM algorithm, except that they are not directly used to compute the motif position distributions. Instead, the algorithm cycles through the sequences individually, removes the site from each sequence, and samples in a new site for that sequence given the current motif distribution. The motif distribution is determined by the current set of sites in all of the other sequences, so the new site is effectively sampled from a conditional distribution. This conditional distribution is the probability of the data given the alignment. Unlike the EM algorithm, the Gibbs Sampler is more likely to explore a larger portion of the sample/parameter space because it does not always follow a direct path to a maximum likelihood estimate. EM algorithms have a tendency to find local maxima rather than global maxima. At the end of Gibbs Sampling, the motif and sampled sites which maximize a statistic quite similar to the information content of the CONSENSUS algorithm are taken as the solution.

2.1.3.2 The Gibbs Motif Sampler

In 1995, the Gibbs Site Sampler algorithm was extended to allow for multiple motifs, multiple sites per sequence, and gaps in motifs (Liu *et al.*, 1995; Neuwald *et al.*, 1995). This incarnation is known as the Gibbs Motif Sampler. Mathematically, the Gibbs Motif Sampler treats all of the sequences as one very long sequence. Each position in the long sequence fits either a motif distribution or the background distribution. This model is similar to the model employed by the MEME algorithm, but it does not impose the constraint of a single motif distribution. At each position, the probabilities that the position belongs to the background model or begins a site belonging to a motif model are calculated, and one of these models is sampled and updated accordingly. Once again, these conditional probabilities are the probabilities of the data given the current alignment. If a position already contains the start of a site, the algorithm removes the site from its motif before performing the sampling. The Gibbs Motif Sampler algorithm also allows gaps, or unconserved positions, in a motif by incorporating a "fragmentation" model. The fragmentation model assumes that a motif has positions which may be turned on or off. Positions which are off do not contribute to the product multinomial probabilities. Positions are turned on and off by another Gibbs Sampling strategy to explore several fragmentation distributions. Finally, the Gibbs Motif Sampler also introduced a method for probabilistically evaluating the complete alignment, accounting for the distribution at every position in the input data. This evaluation is the *a posteriori* probability. Essentially, this probability represents the probability of the alignments given the data. This is calculated

by applying Bayes's Rule to the conditional distributions used in the main sampling step. The maximum *a posteriori* probability, also known as the "MAP", discovered during the alignment sampling process is taken as the best solution.

2.1.3.3 The Gibbs Recursive Sampler

In 1999, Liu, Neuwald, and Lawrence laid the groundwork for a more robust Gibbs Sampler (Liu *et al.*, 1999). Known as the Gibbs Recursive Sampler, this new algorithm outlined a "propagation model" in which information about the conditional probability distributions is recursively propagated to allow for efficient sampling. Given the sites in all sequences but the current one, the algorithm will sample sites in the current sequence. Assuming we know the number of sites that we want to sample in the current sequence, the algorithm calculates the probability that the first site (of any motif type) starts in each position. After the first pass, the algorithm repeats to calculate the probability that the second site begins in any allowable position, given the location of the first site. The algorithm continues to propagate these conditional probabilities until they have been enumerated for all possible site placements. The algorithm then samples in the reverse order, beginning by sampling the last site (and its corresponding motif), and proceeding until it samples the first site. The advantage of this algorithm is that it allows the statistical model to include information about the spacing between sites and the linear ordering of sites by motif type. For example, the algorithm can specify probabilities for scenarios such as "a site of motif type A is between two sites of motif type B" (Thompson *et al.*, 2004). The implementation of the Gibbs Recursive Sampler was also enhanced with the ability to sample the number of sites to be sampled from a given sequence, a necessary feature (Thompson *et al.*, 2003).

2.1.3.4 Other Gibbs Sampling Algorithms

BioProspector (Liu *et al.*, 2001) and AlignACE (Roth *et al.*, 1998) are alternative implementations of the Gibbs Motif Sampler.

2.1.4 Comparison of Algorithms

In 2000, Workman and Stormo introduced a neural network algorithm called ANN-SPEC for motif discovery and performed a quantitative comparison between ANN-SPEC, the Gibbs Site Sampler, CONSENSUS, and MEME (Workman & Stormo,

2000). Their comparison is interesting because it highlights the importance of a robust model for the sequence background composition. They concluded that both ANN-SPEC and the Gibbs Sampler outperform CONSENSUS and MEME when sequence background composition is random. When they constructed a data set with non-random background composition, ANN-SPEC outperformed the Gibbs Sampler. The authors noted that their version of the Gibbs Sampler was unable to learn the background distribution. This has been addressed by allowing the Gibbs Sampler to use sequence and position specific background models (Liu & Lawrence, 1999).

2.2 Algorithms for Inferring Phylogenetic Relationships

There are several algorithms to infer phylogenetic relationships between species from sequence data. Since we will primarily be concerned with computing probabilities of DNA sequence data given a tree, we will focus on that aspect of these algorithms. It is important to remember that constructing the tree, determining its topology and branch lengths, is another very important problem which will only be briefly addressed here.

2.2.1 Mathematically Modeling Evolution

Jukes and Cantor proposed one of the earliest nucleotide substitution models for phylogeny (Jukes & Cantor, 1969). Their model assumed that all nucleotides in a DNA sequence were equally likely to mutate, and nucleotides mutated to any other nucleotide with equally likely probabilities. It is very useful to represent this as a *transition rate matrix*. The Jukes & Cantor transition rate matrix is:

$$\begin{bmatrix} -1 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & -1 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & -1 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & -1 \end{bmatrix} \tag{2.1}$$

In equation 2.1, the rows represent an existing nucleotide, and the columns represent the nucleotides to which it may mutate. The numbers where the rows and columns intersect are the probabilities of that mutation. The number along the diagonals ensure that each row sums to 0. A rate matrix can be used to compute a transition matrix, as shown in section 4.2. Transition matrices in phylogeny are also referred

to as *substitution matrices*. Mathematically, this model is useful because these matrices represent a *stationary Markov process* which can be manipulated to simulate change over time and to allow the nucleotide composition of sequences to converge to reasonable values. After infinite time, given the rate matrix above, the sequence composition is expected to be uniformly random.

2.2.2 Evolutionary Markov Processes

As stated earlier, evolution is mathematically considered a *stationary Markov process*, which means that the probability of a nucleotide mutating is only dependent on its present *state*. The four types of nucleotides in DNA are considered the state of the nucleotide. Given infinite time, the nucleotides will converge to an *equilibrium distribution* because the process is stationary and irreducible. 16S rRNA genes change very little over time, so it is reasonably safe to assume that their sequences are near an equilibrium distribution. The same is also true for any other highly conserved sequence, such as the DNA binding sequence for a transcription factor.

Another interesting aspect of Jukes & Cantor's substitution matrix is that it is reversible. Time can flow forward or backward to converge on the equilibrium distribution, which implies we cannot distinguish ancestors from descendants. Lanave *et al.* explored these reversible substitution processes to develop a method for determining the rate of mutation (Lanave *et al.*, 1984). They also defined evolutionary distance as the expected number of mutations per sequence position between two sequences. Later, Yang showed that reversible substitution matrices generally provided the best models (Yang, 1994*a*).

2.2.3 Felsenstein's Algorithm

While investigating phylogenetic inference using amino acid sequences, Neyman proposed a similar, reversible substitution mechanism which could be used with a maximum likelihood (ML) approach to predict the most likely phylogenetic tree (Neyman, 1971). Specifically, Neyman illustrated how to develop a probabilistic model for three species. Felsenstein extended this model to a tree with an arbitrary number of species and provided an algorithm for finding the most likely tree (Felsenstein, 1981).

Felsenstein's Algorithm is based on the ability to compute the likelihood of observed nucleotide sequences given a phylogenetic tree. Felsenstein assumed that the equilibrium distribution of nucleotides will be the prior probabilities of each type of nucleotide, which may be estimated from the overall nucleotide composition of the

sequences. With this knowledge, one can calculate the likelihood of the sequence data given the tree with a post-order tree traversal. This traversal starts at the leaves of the tree and builds up to the root. The post-order tree traversal greatly simplifies computation, and it is generic enough to work with a variety of other assumptions about the equilibrium distribution. More details of the actual computation are provided in section 4.3.1.

Felsenstein also showed that the position of the root in the tree is actually not important under this reversible Markov process model. Indeed, the flexibility to position the root allows an efficient optimization algorithm for constructing the tree.

2.2.4 Constructing a Phylogenetic Tree

To begin constructing a phylogenetic tree, we need a topology. Felsenstein calculates that there are $\frac{(2n-5)!}{(n-3)!2^{n-3}}$ possible unrooted trees (Felsenstein, 1981). Obviously, an exhaustive search algorithm is not feasible. Felsenstein proposes the relatively simple search strategy of examining the placement of species on the tree in successive order. Starting with a tree of $(n-1)$ species, the algorithm looks at all $2n-5$ possible ways to add the n^{th} species. One at a time, each branch length is adjusted to locally maximize the likelihood of the tree. The likelihood of the tree is calculated for each way, and the topology with the maximum likelihood is taken. The process is repeated until all species are on the tree. Since the order of adding species to the tree affects the topology, it may be worthwhile to explore several different orderings. Felsenstein claims that the topology will be identical for "extremely self-consistent data." This greedy algorithm does not guarantee that the most likely topology or tree will be found, but it produces good results.

2.2.5 Refining Felsenstein's Algorithm

DNA nucleotides are classified as either *purines* or *pyrimidines*; adenine and guanine are purines, and thymine and cytosine are pyrimidines. Mutations in DNA can be classified as either *transitions* or *transversions*. Transitions occur when a purine mutates to the other purine or a pyrimidine mutates to the other pyrimidine. Transversions occur when a purine mutates to a pyrimidine or a pyrimidine mutates to a purine. This distinction is important in phylogeny because transitions are more likely to occur naturally than transversions. Kimura proposed extending the substitution model of Jukes and Cantor to account for this (Kimura, 1980). Kimura added two parameters to the model: the rate of occurrence per unit of time for transitions and the rate for

transversions. Later, Hasegawa, Kishino, and Yano added these parameters to the model in Felsenstein's algorithm (Hasegawa *et al.*, 1985).

Several years later, Yang explored removing the assumption that all positions in sequence data evolve at the same rate over time. Yang discovered that a gamma distribution modeled the rates well, but the computational overhead was quite expensive (Yang, 1993). To improve performance, Yang used a discrete approximation of the gamma distribution (Yang, 1994*b*). Yang's approximation equally divided the gamma distribution into a fixed number of rates with equal probability. The mean of each region was chosen as the rate for that region of the distribution. This method was useful and performed well, but it did not account for correlation between rates of evolution at adjacent sequence positions. Yang approached the correlation problem by using Hidden Markov Models (Yang, 1995). Independently, Felsenstein and Churchill developed another Hidden Markov Model approach (Felsenstein & Churchill, 1996). For an overview of these and other methods, please see Felsenstein's review article (Felsenstein, 2001) or *Inferring Phylogenies* (Felsenstein, 2003).

2.2.6 Other Mathematical Models of Evolution

2.2.6.1 The Links Model

Thorne, Kishino, and Felsenstein also explored the application of phylogeny to the alignment of two sequences (Thorne *et al.*, 1991). They imagined sequences as containing *links* from one nucleotide to the next. These links were subject to a statistical birth-death process which would cause insertions or deletions in the sequence. Since there is no way to differentiate an insertion from a deletion between two sequences, insertion/deletion events are called *indels*. Individual nucleotides are also subject to Felsenstein's earlier substitution process (Felsenstein, 1981). This model became known as the *links* model, and Thorne, Kishino, and Felsenstein later generalized it to model groups of nucleotides as fragments of various sizes with varying evolutionary rates (Thorne *et al.*, 1992).

2.2.6.2 The Tree-HMM Model

Mitchison and Durbin added Hidden Markov Models (HMM) to the assortment of tools for handling insertions and deletions (Mitchison & Durbin, 1995). They called their model *Tree-HMM*, and instead of having only substitution matrices for nucleotides as Felsenstein did (Felsenstein, 1981), they constructed both substitution matrices and separate transition matrices for states of matching nucleotides, inser-

tion of nucleotides, and deletion of nucleotides. They use a dynamic programming algorithm for sequence alignment and a maximum likelihood procedure to fit the alignment and HMM to the tree. In a later paper, Mitchison used Bayesian sampling to simultaneously discover both the tree and alignment (Mitchison, 1999).

2.2.6.3 The Extended Links Model

Holmes and Bruno developed another Bayesian approach to phylogeny leveraging the *links* model (Holmes & Bruno, 2001). Conditioning on a phylogenetic tree, they extended the links model to sample alignments from multiple sequences (as opposed to only two sequences). Holmes and Bruno also discussed the merits of the links and Tree-HMM models and suggested attempting to fuse the two models in the future.

2.3 Phylogenetic Algorithms for Discovering *Cis*-Regulatory Elements

Recently, new algorithms to discover *cis*-regulatory elements with the aid of phylogenetic information have been developed. These algorithms employ a variety of approaches, including optimization, expectation maximization, and Gibbs Sampling. Many of these algorithms are extensions of earlier, non-phylogenetic incarnations.

2.3.1 Optimization Algorithms

2.3.1.1 The PhyloCon Algorithm

The PhyloCon algorithm (Wang & Stormo, 2003*b*) by Wang and Stormo is an extension of the CONSENSUS algorithm (see section 2.1.1). The goal for PhyloCon was to account for conservation of *cis*-regulatory elements across orthologs from related genomes, while the original CONSENSUS algorithm was intended to be used on *cis*-regulatory elements conserved near co-regulated genes in the same species. PhyloCon begins by aligning orthologs using WCONSENSUS (see section 2.4) to produce "profiles". These profiles contain the frequencies of observations of each nucleotide at each position in the alignment of multiple species for a single, orthologous gene. Wconsensus produces suboptimal alignments which are useful for the second step of the algorithm, comparing the profiles. The profiles from regions near genes which are hypothesized or known to be co-regulated are compared in a pairwise fashion via

another alignment step. Instead of aligning sequence observations, the frequency distributions are aligned. Profile alignments are scored using an "average log likelihood ratio" statistic, which is somewhat similar to the information content score used in the original CONSENSUS algorithm. The final step is a greedy algorithm which repeatedly merges pairs of profiles with the highest average log likelihood ratio, discarding pairs below some threshold.

2.3.2 Expectation Maximization Algorithms

2.3.2.1 The OrthoMEME Algorithm

The OrthoMEME algorithm (Prakash *et al.*, 2004) by Prakash *et al.* is an extension of Bailey and Elkan's MEME algorithm (see section 2.1.2. OrthoMEME operates on sets of orthologs from a pair of related species. For example, these sets should be regions upstream of potentially co-regulated genes with orthologs between two species. OrthoMEME only searches for one motif, but it may find zero or more instances of the motif in each pair of orthologs. The parameters for a motif are a traditional position weight matrix, as well as transition probability matrices for each position in the motif. There is also a parameter for the expected number of motif occurrences. The expectation step computes the probability of a motif occurring at each position under the model, while the maximization step updates the model parameters to maximize the likelihood of the data under the model.

2.3.2.2 The EMnEM Algorithm

The EMnEM (Expectation-Maximization on Evolutionary Mixtures (Moses *et al.*, 2004)) algorithm by Moses *et al.* is another extension of the MEME algorithm (see section 2.1). EMnEM operates on aligned orthologs from an arbitrary number of species. EMnEM views this as a sort of collapsed sequence. The probability model for the data is a mixture of two evolutionary models: one for a motif, and one for background. Regions of the sequence belong to either of these models. The evolutionary models are Markov processes of nucleotide substitution, as discussed in section 2.2. EMnEM uses the nucleotide substitution matrix of Jukes & Cantor (see section 2.2.1) with Felsenstein's Algorithm (see section 2.2.3) and a phylogenetic tree to evaluate the model.

2.3.2.3 The PhyME Algorithm

Unlike OrthoMEME and EMnEM, the PhyME algorithm (Phylogenetic Motif Elicitation (Wang & Stormo, 2003*a*)) by Sinha *et al.* is not best described as an extension of MEME. PhyME uses expectation-maximization, but it is not very similar to MEME, OrthoMEME, or EMnEM. The probability model in PhyME integrates two aspects of a *cis*-regulatory element: conservation across species (in orthologs) and overrepresentation within a species (in co-regulated genes). PhyME begins by attempting to align as many regions as possible from all of the input sequences using the MultiLAGAN algorithm (see section 2.4). The result is a set of conserved "blocks" of sequence between each species and a special reference species. All regions of sequence not contained in these blocks are considered unconserved between the species. The sequences are assumed to have been produced by a Hidden Markov Model parameterized by a motif position weight matrix and a background weight matrix. Subsequences which are sampled from the motif weight matrix in aligned blocks are drawn from the evolutionary model of Felsenstein's Algorithm. The expectation-maximization trains the parameters of the weight matrices.

2.3.3 Gibbs Sampling Algorithms

2.3.3.1 The CompareProspector Algorithm

CompareProspector (Liu *et al.*, 2004) is an extension of BioProspector (see section 2.1.3.4) by Liu *et al.* which biases sampling to conserved regions. Before sampling, input sequences are aligned with LAGAN (see section 2.4), and a "window percent identity" (WPID) is computed for all subsequences sharing the length of the target motif. The Gibbs Sampler only samples sites at locations where the WPID is greater than a certain threshold. Site probabilities are also weighted by the WPID.

2.3.3.2 The Li & Wong Algorithm

The Li & Wong algorithm (Li & Wong, 2005) is an extension of the Gibbs Site Sampler (see section 2.1.3.1). This algorithm uses Felsenstein's model of evolution (see section 2.2.3) and assumes that background sequences evolve at a greater rate than regulatory sequences. Substitution matrices are inferred from the maximum likelihood phylogenetic tree for the species of interest. Ancestral *cis*-regulatory elements are assumed to have been drawn from a product multinomial motif, and descendants are assumed to have evolved from this motif through a Markov chain which follows

Felsenstein's model of evolution. The sampler updates the ancestral motif to infer its position weight matrix, and then the sampler updates the *cis*-regulatory elements in the observed sequence data in accordance with the ancestral motif and the model of evolution. This algorithm also samples elements of various widths, but does not handle insertions, deletions, or gaps.

2.3.3.3 The PhyloGibbs Algorithm

The PhyloGibbs algorithm (Siddharthan *et al.*, 2005) by Siddharthan *et al.* is very similar to PhyME, but with several key differences. The most significant difference is that PhyloGibbs uses a Gibbs Sampling approach instead of expectation-maximization. This approach allows PhyloGibbs to search for multiple motifs and multiple *cis*-regulatory elements in each sequence. PhyloGibbs uses the same model of evolution as PhyME, but it restricts itself to a star topology for the phylogenetic tree.

2.4 Algorithms for Global Sequence Alignment

Many of the phylogenetic algorithms discussed in section 2.3 require their input sequences to be aligned. Since all of these algorithms perform "local alignment" of *cis*-regulatory elements, this earlier alignment step may be referred to as "global alignment" of the sequences including the *cis*-regulatory elements. There are many global alignment algorithms, but only the ones reported to have been used with the algorithms in section 2.3 are reviewed here.

2.4.1 The ClustalW Algorithm

ClustalW (Chenna *et al.*, 2003) is one of the oldest and most respected global alignment algorithms. ClustalW is particularly favored because it produces reasonable alignments without requiring the user to tune its parameters. ClustalW begins by performing pairwise alignments for all pairs of sequences to be aligned. The pairwise alignments attempt to minimize the number of mismatches between the sequences. The algorithm then calculates distances between pairs by counting the numbers of mismatches in non-gap positions. These distances form a matrix, and ClustalW uses a neighbor-joining algorithm (hierarchical clustering) to construct a "similarity tree", which is similar to a phylogenetic tree. ClustalW finally uses the tree to combine (align) the closest alignments until one global alignment is achieved.

2.4.2 The Wconsensus Algorithm

WCONSENSUS (Hertz & Stormo, 1999) is an extension of the CONSENSUS algorithm (see section 2.1.1). While the CONSENSUS algorithm attempts to maximize information content in a motif of a specific length, the WCONSENSUS algorithm attempts to maximize "crude information content" in a motif of variable length. "Crude information content" is the traditional information content minus two biases: average information content for the number of sequences in the alignment and a multiple of the standard deviation of the average information content. Allowing the motif length to increase allows the algorithm to increase the crude information content. WCONSENSUS was used with PhyloCon (see section 2.3) to generate many suboptimal, ungapped alignments for the initial profiles (Wang & Stormo, 2003*b*).

2.4.3 The MultiLAGAN Algorithm

The MultiLAGAN algorithm (Brudno *et al.*, 2003) leverages many local alignments to construct a global alignment. MultiLAGAN begins by performing pairwise alignment for all sequences of interest using the LAGAN algorithm. The LAGAN algorithm essentially attempts to find as many short alignments as possible and connects the longest ones at overlapping positions. Using a phylogenetic tree provided by the user, MultiLAGAN then merges the closest sequences into a "multi-sequence" using the pairwise alignment. MultiLAGAN repeats the merge step by aligning the closest multi-sequences until only one multi-sequence remains.

2.4.4 The Threaded Blockset Alignment Algorithm

The Threaded Blockset Aligner (Blanchette *et al.*, 2004) was originally designed to align very long sequences, such as whole genomes. Its operation is somewhat similar to MultiLAGAN. Instead of attempting to find short alignments of subsequences, the Threaded Blockset Aligner attempts to find long alignments of subsequences between each pair of sequences. These alignments are known as "blocks". Overlaps between blocks represent a larger global alignment which may contain some or all of the sequences. Instead of creating one global alignment, the algorithm creates "threads" of these blocks. These threads essentially project the overlapping blocks on to a continuous reference sequence. The allows flexibility to view only relevant, alignable regions instead of forcing one global alignment which includes the less certain alignment regions between overlaps.

Chapter 3

Challenges for a New Method

3.1 Discovering *Cis*-Regulatory Elements in Closely Related Genomes

Several methods exist to use information from multiple genomes to search for *cis*-regulatory elements (McCue *et al.*, 2001; McCue *et al.*, 2002). These methods operate by using Gibbs Sampling to perform multiple local alignment on DNA sequences upstream of orthologous genes. Orthologous genes, or orthologs, are genes with identical biological function present in multiple species which diverged from a common ancestor. Orthologs provide clues to the evolutionary (or "phylogenetic") relationships between species. Probabilistically, observations of data in orthologous sequences are not independent because of these evolutionary relationships, but until recently, multiple local alignment algorithms have considered these observations as independent.

As discussed briefly in section 1.1, one of the greatest problems with statistical analysis of genomes is DNA sequence correlation between closely related species. With our current Gibbs Sampling methods, this problem seems to exhibit itself when comparing genomes at the family level of the taxonomic hierarchy. Correlation was a problem between *Escherichia coli K12* and *Salmonella enterica Typhi* in a previous analysis (McCue *et al.*, 2002), so it seems likely that it will be a problem among the Shewanella species. The phylogenetic tree of figure 3.1 illustrates the phylogenetic relationships between the Shewanella and the rest of the gamma proteobacteria class, as determined using PHYLIP (Felsenstein, 1993) on 16S ribosomal RNA genes from each species.

Chapter 4 introduces a new algorithm for discovering *cis*-regulatory elements in closely related genomes.

Figure 3.1: A Gamma Proteobacteria Phylogenetic Tree

3.2 Analyzing Multiple Genomes Simultaneously

3.2.1 Software Design & Development

Comprehensive open source toolkits for developing bioinformatics applications are available in Perl, Python, Java, and various other languages (Mangalam, 2002; Rice *et al.*, 2000). Most of these toolkits focus on facilitating the interpretation of biological data rather than providing an integrated framework for computationally intensive analysis. We do not believe that any of these open source toolkits will provide adequate functionality and performance for our analysis.

With the deluge of new genome sequences becoming available, scientists need a flexible computational infrastructure to handle intensive computation across massive data sets. As bioinformatics researchers migrate from a competitive environment to a much more collaborative one (Quackenbush, 2003), open standards and open source software implementations are essential. Specifically, there is also demand for an open source implementation of software to search for *cis*-regulatory elements (Jamison, 2003).

Chapter 5 introduces a new software framework to address these issues.

3.2.2 Data Management

Recently, we conducted a project analyzing the genomes of five species of Cyanobacteria, commonly known as blue-green algae (McCue *et al.*, 2006). This analysis project spanned the full genome of each species, focusing on DNA sequence segments near approximately 1600 genes that exist in all five species. These sequence segments were inefficiently stored in separate files, losing some information about their context in their respective genomes. In total, approximately 286,000 files were associated with the project.

Similar analyses have been conducted in the past on sets of up to seven genomes (McCue *et al.*, 2002), and larger analyses will be conducted in the future. As the amount of data and the scope of the analysis grow, relying on huge sets of loosely connected files becomes difficult and error-prone. It is difficult to bring together the correct pieces of data from various portions of the analysis, and it is especially difficult to integrate our information with data from external sources.

Chapter 6 introduces a new database system for managing data from whole genome discovery of *cis*-regulatory elements.

Chapter 4

The Gibbs Phylogenetic Sampler

A new multiple local alignment algorithm which incorporates phylogenetic relationships in its statistical model has the potential to be very useful. We propose to replace the product multinomial model of the Gibbs Recursive Sampler algorithm (Thompson *et al.*, 2003) with a new "product phylogeny" model. Several effective methods of creating phylogenetic trees from DNA sequence data are implemented in the PHYLIP software package (Felsenstein, 1993). Given such a tree, we can compute the probability of the data observed in orthologous species using a variety of methods, some of which are described in section 2.2. We believe that this probability distribution will be significantly more accurate than the simple product multinomial model when analyzing closely related genomes.

Throughout this section, the terms *transcription factor binding site* and *cis-regulatory element* may be used interchangeably.

4.1 The Product Phylogeny Model

Recall the mathematical model of a motif (product multinomial distribution) from section 1.2.2. The Gibbs Sampling algorithms of section 2.1.3 use this motif model to describe transcription factor binding sites. The product multinomial distribution contains the probabilities for each nucleotide at each position within the motif. The probability at each position is determined by the composition of the sites already in the alignment. To evaluate a sequence, one multiplies these probabilities for each position according to the observed nucleotides in the sequence.

Whereas the product multinomial model uses the probabilities of each nucleotide at each position, the product phylogeny model uses the likelihood of each nucleotide at each position being observed in its corresponding species given a nucleotide substi-

tution model. These new likelihoods are calculated using Felsenstein's Algorithm (see section 2.2.3) at each position. To evaluate a sequence, these likelihoods from Felsenstein's Algorithm are multiplied together. This method accounts for the evolutionary dependence between the observed nucleotides of each species.

4.2 The Substitution Model

An implementation of the product phylogeny model can use a stationary Markov process with the substitution matrix of Hasegawa, Kishino, and Yano (Hasegawa *et al.*, 1985). For infinitesimal time, this matrix is given by $I + Qdt$, where

$$Q = \mu \begin{bmatrix} -Q_{AA} & \beta\pi_T & \beta\pi_C & \alpha\pi_G \\ \beta\pi_A & -Q_{TT} & \alpha\pi_C & \beta\pi_G \\ \beta\pi_A & \alpha\pi_T & -Q_{CC} & \beta\pi_G \\ \alpha\pi_A & \beta\pi_T & \beta\pi_C & -Q_{GG} \end{bmatrix} \tag{4.1}$$

$$\begin{aligned} Q_{AA} &= (\beta\pi_T + \beta\pi_C + \alpha\pi_G) \\ Q_{TT} &= (\beta\pi_A + \alpha\pi_C + \beta\pi_G) \\ Q_{CC} &= (\beta\pi_A + \alpha\pi_T + \beta\pi_G) \\ Q_{GG} &= (\alpha\pi_A + \beta\pi_T + \beta\pi_C) \end{aligned}$$

μ sets the average mutation rate, which will be discussed later, I is the 4x4 identity matrix, and dt is a small amount of time. π_i is the equilibrium composition of nucleotide $i\epsilon\{A, T, C, G\}$, which is usually taken from the background composition, and $\sum_{i\epsilon\{A,T,C,G\}} \pi_i = 1$. α is the rate factor for a mutation that is a transition, and β is the rate factor for a mutation that is a transversion. Usually a ratio of transitions to transversions is used to set α and β. If $\alpha = \beta$, the matrix follows the model used in Felsenstein's original algorithm (Felsenstein, 1981). If $\pi_i = \frac{1}{4}\forall i$, the matrix follows the model used by Kimura (Kimura, 1980). In our application, α and β will initially be taken from the literature (Siepel & Haussler, 2004), but later they may be directly inferred from the data. Initially, μ will be chosen using Lanave's method (Lanave *et al.*, 1984), but later we may like to sample it using Yang's method (Yang, 1994*b*).

At time $t = 0$, no mutations have occurred, and equation 4.1 becomes the identity matrix. For a longer time $t > dt$, to calculate the probabilities $P_{ij}(t)$ that the nucleotides have mutated from i to j in time t, we calculate e^{Qt}. To avoid repeatedly

performing this calculation in software, we can analytically solve for $P_{ij}(t)$:

$$P_{ij}(t) = \begin{cases} \pi_j + \pi_j \left(\frac{1}{\Pi_j} - 1\right) e^{-\mu t} + \left(\frac{\Pi_j - \pi_j}{\Pi_j}\right) e^{-\mu t A} & (i = j) \\ \pi_j + \pi_j \left(\frac{1}{\Pi_j} - 1\right) e^{-\mu t} - \left(\frac{\pi_j}{\Pi_j}\right) e^{-\mu t A} & (i \neq j, \text{ transition}) \\ \pi_j \left(1 - e^{-\mu t}\right) & (i \neq j, \text{ transversion}) \end{cases} \tag{4.2}$$

$$\begin{aligned} A &= 1 + \Pi_j \left(\frac{\beta}{\alpha} - 1\right) \\ \Pi_j &= \begin{cases} \pi_A + \pi_G & (\text{if } j \epsilon A, G) \\ \pi_C + \pi_T & (\text{if } j \epsilon C, T) \end{cases} \end{aligned}$$

Equation 4.2 is given on page 438 of *Molecular Systematics* (Hillis *et al.*, 1996). Similar equations are given for the Jukes & Cantor and Felsenstein substitution models. The Gibbs Phylogenetic Sampler will also be able to use these other substitution matrices.

4.3 Sampling *Cis*-Regulatory Elements

Sampling elements follows the procedure given for the Gibbs Recursive Sampler in section 2.1.3 with one significant modification: the product phylogeny model described above replaces the product multinomial model. Specifically, this affects the probability that a transcription factor binding site starts in a particular position. This sampling step is called the *predictive update* step in general Gibbs Sampling.

Let S be the total number of sequences we are examining. Let $R = \{r_1, ..., r_S\}$ be the sequence data, and let $r_{i,j,k}$ be the sequence data for sequence i from position j to position k (a set of nucleotides). Let N be the number of *cis*-regulatory elements corresponding to a particular motif, and let $A = \{a_1, ..., a_N\}$ be the alignment of the potential elements, where a_i represents a pair denoting the sequence and position where the element begins. Let W be the width of the element (number of nucleotides). Let $h(\cdot)$ be a vector function which counts the individual nucleotides by type at a set of positions, and let Θ be the set of all parameters to our probability distribution. Using either the product multinomial or product phylogeny model, we can calculate the probability relative to background that an element begins at position p in sequence

s as:

$$P(a_i = (s,p)|\Theta, R) \propto \prod_{j=1}^{W} \frac{\text{P(position } p+j-1 \text{ belongs to motif } |\Theta)}{\text{P(position } p+j-1 \text{ belongs to background}|\Theta)} \qquad (4.3)$$

With the product multinomial model, $\Theta = \{\overrightarrow{\theta_0}, ..., \overrightarrow{\theta_W}\}$ are the parameters to the multinomial distributions at each position, and $\overrightarrow{\theta_0}$ is a separate multinomial distribution describing all sequence data outside the motif (a.k.a. background). Note that the θ vectors are generally written without the vector marks in the literature, so they are only used here for emphasis. We will borrow from Liu, Neuwald, and Lawrence's notation for raising a vector to a power: $\theta_a^{\theta_b}$ with p-dimensional vectors $\theta_a = \{\theta_{a_1}, ..., \theta_{a_p}\}$ and $\theta_b = \{\theta_{b_1}, ..., \theta_{b_p}\}$ implies $\theta_{a_1}^{\theta_{b_1}} \cdots \theta_{a_p}^{\theta_{b_p}}$ (Liu *et al.*, 1995). The probabilities of equation 4.3 above are:

$$\begin{aligned} \text{P(position belongs to motif)} &= (\theta_j)^{h(r_{s,p+j-1})} \\ \text{P(position belongs to background)} &= (\theta_0)^{h(r_{s,p+j-1})} \end{aligned}$$

With the product phylogeny model, the probabilities of equation 4.3 are calculated using Felsenstein's Algorithm and the substitution model shown in section 4.2. Let Θ remain the multinomial parameters used above. In our first implementation, the multinomial parameters will be used as the equilibrium distributions, so the vectors $\{\theta_0, ..., \theta_W\}$ correspond to $\overrightarrow{\pi}$ in equation 4.1, where $\overrightarrow{\pi} = \{\pi_A, \pi_T, \pi_C, \pi_G\}$. Later, we may need another method to sample equilibrium distributions, which will be discussed in section 4.4. Let T be a phylogenetic tree and its edge lengths as described in section 2.2.4. Let $A_{\{p\}}$ be the nucleotides at position p of each element in the proposed alignment. The probabilities will be the likelihoods of the nucleotides in the proposed alignment given the phylogenetic tree and equilibrium distributions for each position and the background. These become:

$$\begin{aligned} \text{P(position belongs to motif)} &= \text{Felsenstein}(T, A_{\{j\}}, \theta_j) \\ \text{P(position belongs to background)} &= \text{Felsenstein}(T, A_{\{j\}}, \theta_0) \end{aligned}$$

Details of the computation of Felsenstein$(\cdot)$ are described below.

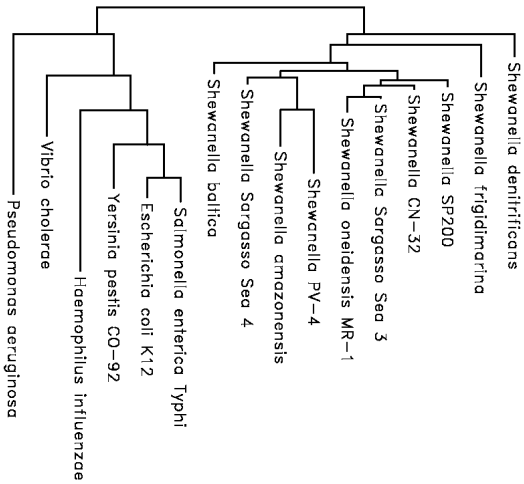


Figure 4.1: A Rooted Phylogenetic Tree of Gamma Proteobacteria

4.3.1 Felsenstein's Algorithm Revisited

Felsenstein's Algorithm uses a post-order tree traversal on a phylogenetic tree to recursively calculate the likelihoods we desire. Each leaf node of the tree has an observed nucleotide in a given species, and each internal node represents a hypothetical ancestral species.

Using Felsenstein's notation (Felsenstein, 1981), let s_i be the nucleotide at node i, and let v_i be the length of the branch leading to node i from its immediate ancestor on the tree. Felsenstein defines $L_s^{(k)}$ to be the likelihood of the subtree connecting node k and all of the leaves which are direct or indirect descendants of node k, given that node k contains nucleotide s. This is best illustrated with a rooted tree, as in figure 4.1, which is equivalent to the unrooted tree shown earlier in figure 3.1. For example, in the rooted tree, let node k be the point where the branch containing *Yersinia Pestis* meets the trunk of the tree. In this case, the tree between node k and its descendant leaves contains *Yersinia Pestis*, *Escherichia coli K12*, and *Salmonella enterica Typhi*. For any node k with children i and j, Felsenstein shows:

$$L_{s_k}^{(k)} = \left(\sum_{s_i} P_{s_k s_i}(v_i) L_{s_i}^{(i)} \right) \left(\sum_{s_j} P_{s_k s_j}(v_j) L_{s_j}^{(j)} \right) \tag{4.4}$$

We also need the base case for the recursion. For a leaf node k:

$$L_{s_k}^{(k)} = \begin{cases} 1 & \text{(if } s \text{ is the observed nucleotide)} \\ 0 & \text{(otherwise)} \end{cases}$$

Recall from section 4.2 that $P_{ij}(t)$ are the nucleotide substitution probabilities. Instead of using Felsenstein's original $P_{ij}(t)$, we will use the one given in equation 4.2. We are interested in the likelihood of an entire tree. Felsenstein showed how to compute this, with the root of the tree labeled as node 0:

$$\sum_{s_0} \pi_{s_0} L_{s_0}^{(0)}$$

For further details and a good example, please see Felsenstein's original article (Felsenstein, 1981).

4.3.2 Recursive Sampling

The recursive sampling procedure proceeds as follows for each sequence:

1. "forward step": calculate probabilities of *cis*-regulatory elements
2. sample number of *cis*-regulatory elements for current sequence
3. "backward step": sample *cis*-regulatory elements in current sequence
4. update motif models to include newly sampled *cis*-regulatory elements

4.3.2.1 Sampling Forward Step

The "forward step" proceeds as follows:

1. For each motif, at each possible position, calculate the ratio of the probability of a *cis*-regulatory element (of that motif) to the probability of background (from the background model).
2. At each possible position, up to some maximum recursion depth, for each motif, calculate the conditional probability of a *cis*-regulatory element (using the ratio from step 1) given that it is preceded by exactly n other *cis*-regulatory elements of particular motifs, where n is the depth of the recursion.

Generally, only one or two motifs is used in a sampling run. The maximum recursion depth is set by the user, but it is often set to only one or two levels. Sampling with one level of recursion is very similar to the Gibbs Site Sampler, without the assumption that one *cis*-regulatory element must be sampled. The run time of step 1 is $\theta(ml)$, where m is the number of motifs and l is the length of the sequence. The run time of step 2 is $\theta(mlr)$, where r is the maximum recursion depth. Note that no sampling occurs in this step.

4.3.2.2 Sampling the Number of *Cis*-Regulatory Elements

Before the sampler completes some amount of "burn-in" iterations, the number of *cis*-regulatory elements in a particular sequence is sampled from a prior distribution. For lack of an informed prior, we use a uniform distribution on $[0, r]$, where r is the maximum recursion depth. After the sampler has completed its burn-in, the number of *cis*-regulatory elements is sampled proportional to the total probability for each level of recursion calculated in the forward step, weighted by an informed prior distribution if provided. These values are computed in the forward step, so the run time for this step is $\theta(r)$.

4.3.2.3 Sampling Backward Step

Let k be the number of *cis*-regulatory elements to sample (which was sampled in the previous step). The *cis*-regulatory elements are now sampled in reverse order from the deepest level of recursion, proportional to the probabilities computed in the forward step. The run time of this step is $\theta(ml)$ since all l positions may be visited during the sampling process.

4.3.2.4 Sampling Model Update Step

The model update step varies with the model for a particular motif. See section 4.4 for discussion on updating product multinomial and product phylogeny models. For both of these models, the run time of this step is $\theta(nw)$, where n is the number of *cis*-regulatory elements and w is the width of the largest motif.

4.3.2.5 Run Time Analysis

Recall that the steps above are run for each sequence. Several to many sequences (generally 5-20) are cycled through in a typical sampling run. The number of cycles through all sequences is determined by the user, but several strategies to determine

when to stop cycling may be employed (see section 4.3.2.6). Let n be the number of sequences and i be the number of total iterations (cycles). Note that "sequences" may include globally aligned sequences as a single sequence. The total run time of the above steps is limited by the forward step, which is $\theta(mlr)$. Considering the input sequences and the cycling, the total run time is $\theta(inmlr)$.

4.3.2.6 Stopping the Sampling Process

Theoretically, given infinite time, the sampler samples sites proportional to their likelihood. In practice, the number of sampling iterations must be bounded. The simplest bound is a constant, where the sampler runs for some fixed number of iterations. The solution may be the final solution sampled, the *maximum a posteriori* (MAP) solution encountered, or some sample of solutions. Work with similar sampling methods in RNA secondary structure prediction (Ding & Lawrence, 2003) has shown that presenting a sample of solutions may be far more useful than a single solution, even if the single solution is optimal by some measure (Ding *et al.*, 2005).

4.4 Updating the Statistical Model

Updating the statistical model is one of the more troublesome parts of the Gibbs Phylogenetic Sampler. In Bayesian Statistics, anything not observed is treated as missing data. Bayesian Statistics also specify prior distributions which give prior information about parameters to a distribution, but these details will be omitted for now. The parameters to the product multinomial model are not observed, but they are a function of the sequence alignment, which is sampled as a random variable from the product multinomial distribution. Specifically, $\theta_j = \Big(h(A_{\{j\}}) + \beta\Big) \,/\, \Big|h(A_{\{j\}}) + \beta\Big|$, which is the posterior mean of the distribution. β is a vector of Bayesian pseudocounts. Obtaining parameters in this manner is known as *collapsing* or *grouping* (Liu, 1994). The parameters to the product phylogeny model (the equilibrium distributions) are not observed, and there does not appear to be a straightforward method to sample them from the product phylogeny distribution. It may be possible to use an importance sampling method to obtain a posterior mean that is useful, but this remains to be investigated. Importance sampling methods for Markov Chain Monte Carlo applications are discussed in *Monte Carlo Strategies in Scientific Computing* (Liu, 2001).

Empirical evidence suggests that the parameters to the product phylogeny model can be reasonably estimated from the posterior mean of the product multinomial

distribution. For improved accuracy, these estimates are weighted by sequence weights (see below). Given a weight w for a particular species, each observation from that species counted by $h(\cdot)$ is counted w times instead of once. A vector of sequence weights need only be computed once for a given tree.

4.4.1 Sequence Weights

We compute "optimal sequence weights that minimize the sum of the variances of the estimators of base frequency parameters for sequences related by a phylogenetic tree" (Newberg *et al.*, 2005). We begin by computing a pairwise distance matrix D from the phylogenetic tree. Recall that each leaf is a species. Also recall that distances along branches of the tree are additive, so determining the pairwise distances for all species is not difficult with a post-order tree traversal:

1. For each node, request the distances to all leaf nodes from its children.
2. After the distances to all leaf nodes from the children have been returned, add the distance between the parent and child to each returned distance.
3. Check all pairs of observed leaf nodes to determine if their distance has been computed yet. If not, their distance is the sum of the distances to each leaf node from the current node (step 2).

With this matrix D, we compute a new matrix C. For each entry $D_{i,j}$, let $C_{i,j} = e^{-kD_{i,j}}$, where k is the rate parameter. Generally, for an alphabet of with a letters, we let $k = \frac{a}{a-1}$. The vector of sequence weights W is now computed as $W = C^{-1}\overrightarrow{1}$, where $\overrightarrow{1}$ is a column vector filled with 1s and with length equal to the number of species. Negative weights are set equal to 0, and the final vector of sequence weights is scaled to preserve the original sum of the weights.

4.5 Evaluating the Predicted *Cis*-Regulatory Elements

As mentioned in section 2.1.3, potential alignments are evaluated by calculating the *a posteriori* probability of the alignment given the data. To do this, we integrate over all possible values of the parameters:

$$P(A|R) = \int P(A|R,\Theta)P(\Theta)d\Theta$$

For the product multinomial model (without prior probabilities for simplicity), this is:

$$P(A, \Theta|R) \propto \theta_0^{h(R_{\{-A\}})} \prod_{j=1}^{W} \theta_j^{h(A_{\{j\}})}$$

We integrate over all Θ to obtain:

$$P(A|R) \propto \frac{\Gamma\left(h(R_{\{-A\}})\right)}{\Gamma\left(\left|h(R_{\{-A\}})\right|\right)} \times \prod_{j=1}^{W} \frac{\Gamma\left(h(A_{\{j\}})\right)}{\Gamma\left(\left|h(A_{\{j\}})\right|\right)}$$

Above, $R_{\{-A\}}$denotes the background, all nucleotides not in a predicted *cis*-regulatory element. Using Liu, Neuwald, and Lawrence's notation, $\Gamma(\cdot)$ on a vector $V = \{v_1, ..., v_n\}$ indicates $\Gamma(v_1) \cdots \Gamma(v_n)$, and $|v| = |v_1| + \cdots + |v_n|$ (Liu *et al.*, 1995).

For the product phylogeny model, there does not appear to be an analytical method to perform the required integration; however, we can use importance sampling to approximate this integral. To begin, we want to integrate

$$P(A, \Theta|R) \propto \prod_{j \epsilon R_{\{-A\}}} \text{Felsenstein}(T, R_{\{j\}}, \theta_0) \times \prod_{j=1}^{W} \text{Felsenstein}(T, A_{\{j\}}, \theta_j)$$

over all Θ, where $R_{\{j\}}$indicates the nucleotides with position j in each sequence.

To perform integration by importance sampling, we need an integrable function $g(\theta; S)$ (where S is a set of aligned nucleotides such as $R_{\{j\}}$ or $A_{\{j\}}$) which approximates $\text{Felsenstein}(T, S, \theta)$. Since Felsenstein's Algorithm is essentially replacing a multinomial variable in the Gibbs Phylogenetic Sampler, let $g(\theta; S) = \theta^{h(S)}$.

$$\begin{aligned} \int \text{Felsenstein}(T, S, \theta) d\theta &= \int \frac{\text{Felsenstein}(T, S, \theta)}{g(\theta; S)} g(\theta; S) d\theta \\ &= E_{g(\theta;S)} \left[\frac{\text{Felsenstein}(T, S, \theta)}{g(\theta; S)} \right] \times \int g(\theta; S) d\theta \end{aligned}$$

Note that we can integrate $g(\theta; S)$:

$$\int g(\theta; S) d\theta = \frac{\Gamma(\theta)}{\Gamma(|\theta|)}$$

Finally,

$$P(A|R) = E_{g(\theta)} \left[\prod_{j \epsilon R_{\{-A\}}} \frac{\text{Felsenstein}(T, R_{\{j\}}, \theta)}{g(\theta; R_{\{j\}})} \right] \times \frac{\Gamma\left(h(R_{\{-A\}})\right)}{\Gamma\left(\left|h(R_{\{-A\}})\right|\right)}$$

$$\times \prod_{j=1}^{W} \left\{ E_{g(\theta)} \left[\frac{\text{Felsenstein}(T, A_{\{j\}}, \theta)}{g(\theta; A_{\{j\}})} \right] \times \frac{\Gamma\left(h(A_{\{j\}})\right)}{\Gamma\left(\left|h(A_{\{j\}})\right|\right)} \right\}$$

Through importance sampling, the above expectations can be estimated by drawing a large number of samples of θ from $g(\theta; S)$ and taking the average.

Chapter 5

Bioinformatics Research Application Software System (BRASS)

5.1 The System Architecture

The Gibbs Phylogenetic Sampler is essentially a combination and extension of existing algorithms applied to a type of data set where existing methods are not expected to yield results. It is natural to desire a generic solution in anticipation of future problems. To implement the Gibbs Phylogenetic Sampler, we developed a flexible software system, particularly well suited to manipulating large quantities of sequence data, which facilitates efficient statistical computation.

We have created a new open source software framework as part of our solution. Our framework provides an object oriented, high performance, scalable, extensible C++ (Stroustrup, 2000) application programming interface (API) for creating a new generation of genomic scale analysis software. To address the data management issues, the framework is designed to facilitate interactions with a relational database management system which is described in detail in chapter 6.

5.2 Software Design

A software framework "provides an integrated set of domain-specific structures and functionality based on patterns" (Schmidt *et al.*, 2004). More specifically, a software framework is composed of *components* and *patterns* (Johnson, 1997). A component

is a reusable piece of software code. As described in *Design Patterns*, patterns are "descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context" (Gamma *et al.*, 1995). A framework provides a context for patterns as well as components which facilitate the implementation of patterns. Additionally, a framework may be a component in a larger system. Frameworks span both the abstract design and concrete implementation of software systems. For more information on software frameworks, please see the Fayad, Schmidt, and Johnson books (Fayad *et al.*, 1999*a*; Fayad *et al.*, 1999*b*).

The domain for our framework is statistical biological sequence analysis. Our main application is a Gibbs Sampler, and this can be described generically as a pattern. In Bayesian statistics, all Gibbs Samplers sample random variables from conditional distributions to characterize missing data from observed data. The general process is:

1. Observe data and define missing data.
2. Assume that missing data fits a specific type of probability distribution (the model).
3. Randomly guess values for the missing data and parameters to the model.
4. Sample missing data given the model.
5. Sample the model given the data.
6. Calculate probability of predicted missing data.
7. Repeat steps 4-6 until convergence criteria are met.

From the above pattern, we can infer some very generic components that we will need:

- Data
- Model
- Sampler

These components interact together to perform Gibbs Sampling. The object model, shown in appendix A, attempts to facilitate the implementation of this pattern. Specifically, the BRASS::IO namespace contains the data components. The BRASS::Math namespace contains the model components. The BRASS::Samplers

namespace contains the sampler components. The BRASS::Trees namespace contains generic classes for handling tree data structures. The BRASS::Util namespace contains wrappers for Standard Template Library (STL) classes. The individual classes are described below.

5.2.1 Software Design Goals

BRASS was designed with the following software design goals: BRASS should...

- significantly ease development of software for statistical bioinformatics applications.
- provide efficient, easily reusable, object oriented source code.
- be easy to use in a grid computing environment.
- require external libraries only when the development cost of using them is significantly less than the development cost of implementing the necessary functionality.

5.3 BRASS Classes

This section documents the essential design and functionality of each class in the BRASS library. It is intended to describe core design decisions, not implementation details.

5.3.1 BRASS::IO

UML diagrams for BRASS::IO are shown in figures A.1, A.2, and A.3.

5.3.1.1 Letter (a.k.a. basic_Letter)

This class provides a generic mechanism for using objects or primitive data types as symbols in an alphabet. basic_Letter provides information about the storage of letters, usually from a sequence. Public members LetterType and LetterSymbol export this information. LetterType (or letter_t) is the type of the numeric representation of a letter. For more complicated cases, such as paired letters treated as a single letter (as in some aligned data), it makes sense to create a class to represent this. Such a class will need to provide conversion from its representation to a number of type

LetterType. LetterSymbol is a primitive or class which is the usual representation of a letter. There should be a 1:1 mapping of LetterSymbol to LetterType. For example, with DNA, it is convenient to use ASCII characters as LetterSymbols and their numeric equivalents as LetterTypes.

5.3.1.2 AlignedLetter

AlignedLetter provides a mechanism for storing all of the observed letters in an alignment as a single letter. AlignedLetter implements the STL Container concept to allow convenient access to the individual letters in the AlignedLetter.

5.3.1.3 Alphabet

An Alphabet is a set of symbols which describe individual positions in a generic sequence. Specifically, it is a set of letters which have some meaning together. The Alphabet class is abstract.

5.3.1.4 AlphabetASCII

AlphabetASCII is an abstract class derived from Alphabet. AlphabetASCII is essentially a tag to limit letters in the alphabet to ASCII characters.

5.3.1.5 AlphabetDNA

AlphabetDNA is a concrete class derived from AlphabetASCII which provides an alphabet comprised of the nucleotides present in DNA sequences. This class also provides letters for gaps in sequences, masks in sequences, and unknown nucleotides.

5.3.1.6 AlphabetRNA

Similar to AlphabetDNA, AlphabetRNA is derived from AlphabetASCII and provides an alphabet comprised of the nucleotides present in RNA sequences.

5.3.1.7 AlphabetProtein

Similar to AlphabetDNA, AlphabetProtein is derived from AlphabetASCII and provides an alphabet comprised of the amino acids in a protein's primary structure sequence.

5.3.1.8 SequenceData (a.k.a. basic_SequenceData)

The basic_SequenceData class provides a mechanism for storing and accessing a set of sequences. This class also implements the STL Container concept for usage convenience. SequenceData is typically used within the BRASS library for generic sequences and generic, globally aligned sequences (as basic_SequenceData<SequenceAlignment>, a.k.a. SequenceDataAligned).

5.3.1.9 Sequence (a.k.a. basic_Sequence)

The basic_Sequence class provides a mechanism for storing and accessing a single, generic sequence.

5.3.1.10 SequenceAlignment

The SequenceAlignment (a.k.a. basic_Sequence<AlignedLetter>) class provides storage for a set of aligned sequences, such as orthologs.

5.3.1.11 SequenceFeature

A SequenceFeature is a point of biological interest along a sequence. An example is a binding site for a particular transcription factor. This class only describes the information about the feature in the data (sequence and position), not any mathematics which may describe the feature in some other way. It may be associated with a mathematical model.

5.3.1.12 SequenceFeatureSet

A SequenceFeatureSet is a group of features. This class implements the STL Container concept to provide convenient access to the individual SequenceFeatures.

5.3.1.13 SequenceDataFactory

The SequenceDataFactory class contains methods to create new instances of Sequence and SequenceData from files. For example, this class is used to read sequence data from FASTA files.

5.3.2 BRASS::Math

Figure A.5 illustrates an overview of the BRASS::Math namespace in UML. Additional details are shown in figures A.6, A.7, A.8, and A.9.

5.3.2.1 LocalAlignment

LocalAlignment is a local alignment of features in BRASS::IO::Sequence objects. For example, these may be transcription factor binding sites. Note that in globally aligned data, these are actually alignments of alignments since a SequenceAlignment is essentially a collapsed form of close sequences.

5.3.2.2 SequenceFeatureModel

SequenceFeatureModel is an abstract class (*i.e.* interface) representing a mathematical model of a sequence feature (either foreground or background). These are often called motifs, especially for foreground models. An example model for an alignment is a product multinomial motif. Another example is a generic background model which may describe a piece of sequence. See SequenceForegroundModel and SequenceBackgroundModel.

5.3.2.3 SequenceForegroundModel

SequenceForeground model is an abstract class (*i.e.* interface) derived from SequenceFeatureModel which represents a mathematical model of a foreground sequence feature (such as an alignment). These are often called motifs. An example model for an alignment is a product multinomial motif. Unlike background models, foreground models are sampled from the data. See MotifProduct.

5.3.2.4 MotifProduct

MotifProduct is a concrete class derived from SequenceForegroundModel which implements a generic multiplicative motif for locally aligned sequences. Product multinomial and product phylogeny motifs are example uses of MotifProduct. These instantiations are implicit by the type of "columns" in the MotifProduct. Each column is the model for a specific position in the multiplicative motif. This allows mixing and matching of columns. A better example is the use of "null" columns to represent positions in a motif which are unmodeled. See MotifColumn for more detail.

5.3.2.5 MotifColumn

MotifColumn is an abstract class (*i.e.* interface) for a column of a product style motif (MotifProduct). Examples of column types may be multinomial, as in product multinomial, or phylogeny, as in product phylogeny. Generically, this interface is simple enough to describe any distribution which can be evaluated and may need a corresponding model to be updated. See MotifColumnMultinomial, MotifColumnPhylogeny, and MotifColumnNull for examples.

5.3.2.6 MotifColumnMultinomial

MotifColumnMultinomial is a concrete class derived from MotifColumn. This class represents a multinomial distribution. Letters from an Alphabet are the possible outcomes. This is used as a column in a product multinomial distribution. This class also includes Bayesian pseudocounts. See section 4.3.

5.3.2.7 MotifColumnNull

MotifColumnNull is a concrete class derived from MotifColumn. This class represents an unmodeled position in a multiplicative motif. The probability of any outcome/observation is 1.

5.3.2.8 MotifColumnPhylogeny

MotifColumnPhylogeny is a concrete class derived from MotifColumn. This class implements Felsenstein's Algorithm for motif positions in a multiplicative motif. MotifColumnPhylogeny runs Felsenstein's Algorithm on alignments of globally aligned data (ie. SequenceAlignment). See FelsensteinAlgorithm.

5.3.2.9 SequenceBackgroundModel

SequenceBackground model is an abstract class (*i.e.* interface) derived from SequenceFeatureModel. Similar to SequenceForegroundModel, this class represents features as background models. Unlike foreground models, background models are held fixed and not sampled. This interface should be implemented by all background models. See SequenceBackgroundModelComposition, SequenceBackgroundModelPhylogenyComposition, SequenceBackgroundModelPhylogenyUnified, SequenceBackgroundModelUnified, SequenceBackgroundModelUnified, SequenceBackgroundModelUniform.

5.3.2.10 BackgroundProduct

BackgroundProduct is a concrete class derived from SequenceBackgroundModel. It implements a generic multiplicative background for locally aligned sequences. Product multinomial and product phylogeny backgrounds are example uses of BackgroundProduct. These instantiations are implicit by the type of columns in the BackgroundProduct. The primary difference between BackgroundProduct and MotifProduct is that the model for BackgroundProduct is fixed, not sampled.

5.3.2.11 SequenceBackgroundModelUniform

SequenceBackgroundModelUniform is a concrete class derived from SequenceBackgroundModel. SequenceBackgroundModelUniform is the simplest background model. This model assumes that all observations are equally likely considering the number of letters in the alphabet of the sequence.

5.3.2.12 SequenceBackgroundModelComposition

SequenceBackgroundModelComposition is a concrete class derived from SequenceBackgroundModel which calculates the background as the composition of observations (letters) in the sequences. The composition provides the parameters to the product multinomial distribution to calculate the probabilities of observations in the sequences.

5.3.2.13 SequenceBackgroundModelUnified

SequenceBackgroundModelUnified is a concrete class derived from SequenceBackgroundModel. SequenceBackgroundModelUnified uses a previously computed position-specific composition as a background model for sequences. The position-specific composition provides the parameters to the product multinomial distribution to calculate the probabilities of observations in the sequences. Currently, the position-specific composition is obtained from the Unified segmentation program (Liu & Lawrence, 1999).

5.3.2.14 SequenceBackgroundModelPhylogenyComposition

SequenceBackgroundModelPhylogenyComposition is a concrete class derived from SequenceBackgroundModel. Similar to SequenceBackgroundModelComposition for sequences, SequenceBackgroundModelPhylogenyComposition uses the composition of

sequence alignments as a background model. Essentially, the composition provides the equilibrium distribution for Felsenstein's Algorithm to calculate the probabilities of observations in the sequence alignments. See SequenceAlignment and FelsensteinAlgorithm.

5.3.2.15 SequenceBackgroundModelPhylogenyUnified

Similar to SequenceBackgroundModelUnified for sequences, SequenceBackgroundModelPhylogenyUnified uses a previously computed position-specific composition as a background model for sequence alignments. The position-specific composition provides the equilibrium distribution for Felsenstein's Algorithm to calculate the probabilities of observations in the sequence alignments. See SequenceAlignment and FelsensteinAlgorithm. Currently, the position-specific composition is obtained from the Unified segmentation program (Liu & Lawrence, 1999). SequenceBackgroundModelPhylogenyUnified is not derived from SequenceBackgroundModel because its role is to provide SequenceDistributions which can be used with BackgroundProduct to create a background model.

5.3.2.16 SequenceDataModelFactory

In Gibbs Sampling, parameters for initial models are drawn at random. SequenceDataModelFactory provides methods to create MotifProduct objects with randomly drawn parameters. The user may specify multinomial or phylogeny MotifColumn types for new MotifProduct objects created by the factory. SequenceDataModelFactory also creates BackgroundProduct objects based on existing distributions, such as sequence composition or a previously computed position-specific background distribution.

5.3.2.17 SequenceDistribution

A SequenceDistribution is a concrete class derived from Sequence. It is a special type of sequence, where each position of the sequence is a probability distribution; *i.e.* each position contains probabilities for each observable letter from the sequence alphabet.

5.3.2.18 SequenceWeights

SequenceWeights is a concrete class which implements a sequence weighting calculation. Specifically, this class provides a method which computes "optimal sequence weights that minimize the sum of the variances of the estimators of base frequency

parameters for sequences related by a phylogenetic tree" (Newberg *et al.*, 2005). This implementation assumes a Jukes-Cantor model. See section 4.4.1.

5.3.2.19 FelsensteinAlgorithm

This class implements Felsenstein's Algorithm (Felsenstein, 1981), a method for computing the probability of observing aligned nucleotides given a phylogenetic tree. It requires a FelsensteinTree and a SubstitutionProcess.

5.3.2.20 FelsensteinTree

FelsensteinTree is a concrete class derived from NewickTree. Specifically, FelsensteinTreeNode is derived from NewickTreeNode, and the tree itself is a BRASS::Trees::Tree<FelsensteinTreeNode>. It is identical to a NewickTree, except each node also contains a substitution matrix. The substitution matrix is computed by FelsensteinAlgorithm (with a SubstitutionProcess) from the alignment data and is used to compute various probabilities of observations across the tree.

5.3.2.21 SubstitutionProcess

Felsenstein's Algorithm allows various models of nucleotide substitution (evolution). SubstitutionProcess is an abstract class (*i.e.* interface) which provides a generic mechanism for creating the substitution matrices used by FelsensteinAlgorithm. This abstraction allows changes to the substitution process to be made without altering FelsensteinAlgorithm. See SubstitutionProcessF81. Alternative substitution processes are described in sections 2.2.5 and 4.2.

5.3.2.22 SubstitutionProcessF81

SubstitutionProcessF81 is a concrete class derived from SubstitutionProcess which implements Felsenstein's original substitution process (Felsenstein, 1981), using the edge length interpretation of Lanave *et al.*

5.3.2.23 SequenceDataModel

SequenceDataModel is a storage class which contains information about the mathematical models used to describe the data in a SequenceData structure. The model needs to know local foreground features, such as locally aligned binding sites, as well as the mathematical models for these features. The models associate directly with

their data, which can be accessed through their accessor methods. SequenceDataModel essentially coordinates data, foreground models, and the background model for a particular analysis.

5.3.3 BRASS::Samplers

Figure A.4 illustrates the BRASS::Sampler classes in UML.

5.3.3.1 DataModelSampler

DataModelSampler is an abstract class to allow generic sampling using a set of models and data. Currently, only Gibbs Samplers implement this interface.

5.3.3.2 GibbsSiteSampler

GibbsSiteSampler is an implementation of a simple Gibbs Sampler on sequence data with corresponding models. This sampler assumes that each sequence contains exactly one instance of one model. Instances of the model are sampled based on their ratio to a background model. This is the sampling method used by Lawrence *et al.* in 1993 (Lawrence *et al.*, 1993). GibbsSiteSampler implements the DataModelSampler interface.

5.3.3.3 GibbsRecursiveSampler

GibbsRecursiveSampler is an implementation of a dynamic programming Gibbs Sampler on sequence data with corresponding models. This sampler allows multiple instances of multiple models in each sequence. The number of instances in each sequence is sampled from the total likelihood, and instances of each model are sampled based on their ratio to background. This is the sampling method described by Liu *et al.* in 1999 (Liu *et al.*, 1999) and used by Thompson *et al.* in 2003 (Thompson *et al.*, 2003). GibbsRecursiveSampler implements the DataModelSampler interface.

5.3.3.4 FrequencySolution

FrequencySolution is a class which counts features in SequenceFeatureSets by sequence and position. With a Gibbs Sampler, this class can provides the information required to construct a frequency solution.

5.3.3.5 GibbsSampler

GibbsSampler is a storage class which coordinates the data, models, and sampler used for a particular analysis.

5.3.4 BRASS::Trees

5.3.4.1 Tree

This class provides a generic tree which allows an arbitrary number of children at any given node. Copy constructors and assignment operators are intentionally declared private and undefined to avoid implicit copying of a Tree. A clone() member function is provided to explicitly copy a Tree.

5.3.4.2 NewickTree (a.k.a. Tree<NewickTreeNode>)

A NewickTree is a Tree which represents a Newick tree as described by Felsenstein (Felsenstein, 1993). Each node has a branch length to its parent and a label. At leaf nodes, the species name is the label.

5.3.4.3 FelsensteinTree (a.k.a. Tree<FelsensteinTreeNode>)

A FelsensteinTree is a NewickTree which also stores transition matrices at each node.

5.3.4.4 NewickTreeFactory

NewickTreeFactory provides a mechanism to read a Newick Tree from a file.

5.3.4.5 NewickTreeInput

NewickTreeInput is a helper class for NewickTreeFactory which uses the Boost Spirit parser to parse a Newick tree from a text file.

5.3.5 BRASS::Util

5.3.5.1 List

List is a simple wrapper class for the STL List class. This wrapper was written to facilitate safely deriving new classes from List.

5.3.5.2 MultiMap

MultiMap is a simple wrapper class for the STL MultiMap class. This wrapper was written to facilitate safely deriving new classes from MultiMap.

5.3.5.3 Vector

Vector is a simple wrapper class for the STL Vector class. This wrapper was written to facilitate safely deriving new classes from Vector.

5.4 Testing & Quality Assurance

Unit tests are small pieces of software written to test other pieces of software. For example, someone might write a piece of software which calculates the area of a circle, πr^2. It is known that the software should produce a value $\pi r^2 \pm \varepsilon$, where ε is some acceptable amount of error. If the software ever produces a value outside this interval, it is incorrect. A unit test can be written to ensure that the value is in this range. The unit test simply returns a value of PASS or FAIL. The unit test becomes very valuable as the code changes over time. For example, a second software developer might (stupidly) change the code for the area of the circle to compute the circumference. Any code which calls the area computation code could be affected by this change. When the unit test is run, it will detect that the area is no longer accurate.

Unit tests in the BRASS source code ensure that code changes do not produce unseen, harmful side effects such as those described above. Unit tests have been written for all major numeric calculations in BRASS, including Felsenstein's Algorithm, sequence weights, and product multinomial calculations. Each unit test determines if a calculation performed by BRASS matches a known, good result. The known results were computed without the use of the software since this would invalidate the results. Each time the library is recompiled, the full suite of unit tests may be run to ensure that all results remain as expected. Unit tests also exist for the Gibbs Sampling code to ensure that the samplers converge consistently on correct alignments.

Gibbs Site Sampling and Gibbs Recursive Sampling have each been tested with both synthetic data and real data known to contain actual *cis*-regulatory elements. In all cases, the samplers converge on the correct alignments. See chapter 7 for additional detail about verifying the results of the BRASS Gibbs Recursive Sampler code.

5.5 Dependencies

The BRASS library depends on only two external libraries: Boost (Boost.org, 2004) and Xerces (The Apache Software Foundation, 2005). BRASS makes use of the Boost library for random number generation, linear algebra functionality, various other mathematical functions, memory management, and unit testing. Xerces provides XML (The World Wide Web Consortium, 2006) parsing support. Future incorporation of an XML parser into Boost may remove the need for Xerces. Currently, BRASS is implemented using the stable tree of Boost 1.32 and Xerces 2.7.0.

The BRASS source code is commented using a format suitable for generating an application programming interface (API) reference manual for developers. Doxygen (van Heesch, 2005) can be used to generate the reference directly from the source code.

5.6 Supported Platforms

The framework is targeted for the g++ compiler on Solaris (SPARC and x86) and Linux (x86). It has been tested with gcc 3.3.6, Solaris 9, and Gentoo Linux (kernel 2.6.x).

Chapter 6

BRASS Database System

To search for transcription factor binding sites, we often analyze sequences from several related species. Recently, we conducted a project analyzing the genomes of five species of Cyanobacteria (McCue *et al.*, 2006), commonly known as blue-green algae. This analysis project spanned the full genome of each species, focusing on DNA sequence segments near approximately 1600 genes that exist in all five species. These sequence segments were inefficiently stored in separate files, losing some information about their context in their respective genomes. The analysis also produced approximately 68,000 new files of results from several programs that perform specific steps of the analysis.

Similar analyses have been done in the past on sets of up to seven genomes (McCue *et al.*, 2002), and larger analyses will be done in the future. As the amount of data and the scope of the analysis grow, relying on huge sets of loosely connected files becomes difficult and error-prone. It is difficult to bring together the correct pieces of data from various portions of the analysis, and it is especially difficult to integrate our information with data from external sources.

6.1 Planning the System

6.1.1 The Users and Their Needs

A database system for bioinformatics sequence analysis will generally serve three types of users: *biologists*, *mathematicians*, and *computer scientists*. For the purposes of this system, biologists include molecular biologists and molecular biochemists, and mathematicians include statisticians.

Biologists are primarily concerned with the biological meaning of the data sur-

rounding our predictions of transcription regulatory mechanisms. They specify the input for our genomic analyses, and they interpret the output. Biologists? interpretations involve connections between various analyses and external sources which may partially be represented as a database query. Integration of information is essential to an effective analysis of a bioinformatics problem.

Mathematicians are concerned with the mathematical details used in the analysis process. In our case, they focus on statistical parameters and variations between sampling runs. Database queries should aggregate this information for more convenient interpretation. For example, a mathematician may want to analyze the variance of an estimate of a parameter to a probability distribution under a certain set of conditions. In an ideal scenario, a mathematician could query a historical database with the conditions of interest and pull out the data they require.

Computer scientists are concerned with the software behind the analysis process. In the case of this database system, the computer scientist will want a conceptual schema and application programming interface (API) that facilitates easily connecting our current software tools to the system while allowing a scalable interface for future tools to be integrated. Obviously, software must populate the database and provide an interface for other users to make meaningful queries, but the design of this system as a whole must be open to interface with new generations of analysis software.

6.1.2 Existing Data and Tools

The database system is useless without data. To best demonstrate the benefits of the system, the system should be designed to leverage results from our existing data and tools. In particular, our current cross-species comparison analysis process employs two primary tools: the Gibbs Recursive Sampler (Thompson *et al.*, 2003) and the Bayesian Motif Clustering (BMC) (Qin *et al.*, 2003) application.

The Gibbs Recursive Sampler takes genomic sequences and statistical parameters as input to produce potential transcription factor binding sites and corresponding mathematical models as output. The database should track both the input and output to maintain a clear record of experiments.

The Bayesian Motif Clustering application takes mathematical models of transcription factor binding sites (often from the Gibbs Recursive Sampler) and statistical parameters to produce clusters of potentially co-regulated genes. The database should track the input and output, and it should maintain relationships between clustered models from the BMC application and predictions of sites from the Gibbs Recursive

Sampler.

The Cyanobacteria analysis project mentioned earlier is a good test case for the database system because it encompasses tens of thousands of loosely connected files. Importing the data and our analysis results into the database demonstrates the potential of the system to integrate information and make it more easily accessible.

6.1.3 Selecting the Database Management System

The enterprise level scope of this system requires an enterprise database management system (DBMS). The DBMS of choice should have strong support for relational integrity constraints, transactions, and stored procedures. Relational integrity is extremely important when integrating information from various sources. In the case of this system, it is very useful to have the database verify that new records reference valid old records. Many entries to the database require updating multiple tables. By using transactions to update these tables, the whole process can be undone if updating a particular table fails. This greatly helps ensure the consistency of data in the system. This system does not have any specific need to use stored procedures at this time, but it seems quite possible that they will be useful in the future. For example, stored procedures could be used to validate input from web users.

6.2 System Design & Analysis

6.2.1 Perspectives

The data in the system has been partitioned into various domains to facilitate organization. To allow for future expansion, we would like to have one domain for each application using the software framework, with separate domains for data which is application independent.

The first implementation of this database system captures input and output from our Gibbs Sampler and BMC applications. Data from external sources is also shared between these applications. To model these requirements, the applications and the shared data comprise the three perspectives toward the database. A web interface can be developed to query the system and present relevant data from each perspective or a combination of perspectives.

6.2.2 System Data

Data to be captured is based on our current Gibbs Sampler and BMC reports, as well as input data to these applications. The shared data (GenomeSharedData), Gibbs Sampler (Gibbs), and BMC perspectives capture the following data:
Genome Shared Data Domain

ortholog_promoter The *ortholog promoter* represents the location in a genome, orientation, and confidence level of an ortholog relationship between a pair of species.

intergenic_location The *intergenic location* represents the location of intergenic sequence upstream of a particular gene.

sequence_data The *sequence data* represents the actual DNA sequence and location within a genome.

Gibbs Domain

gibbs_run The gibbs_run represents the input to and output of a particular run of the Gibbs sampler application.

gibbs_analysis The gibbs_analysis represents a set of Gibbs sampler application runs grouped together by a user.

motif The motif represents the position weight matrix mathematical model of a transcription factor binding site.

position The position represents the parameters for one of the positions of a motif.

binding_site The binding_site represents the location of a particular transcription factor binding site.

binding_site_set The binding_site_set represents a set of binding sites which is generally related to a motif.

BMC Domain

bmc_run The bmc_run represents the input to and output of a particular run of the BMC application.

bmc_motif The bmc_motif represents a mathematical model of a transcription factor binding site. It may be related back to a regulon and/or motif from Gibbs.

bmc_cluster The bmc_cluster FD represents bmc_motifs that have been clustered together by the BMC application.

6.2.3 Semantic Data Model

The relations and functional dependencies for the domains are provided below.

Genome Shared Data Domain Relations & Functional Dependencies

- ortholog_promoter: {genome, gene, orthologous_species, blast_e_score, reverse_complement}
  genome, gene, orthologous_species → blast_e_score, reverse_complement
- intergenic_location: {genome, gene, start_coordinate, end_coordinate }
  genome, gene → start_coordinate, end_coordinate
- sequence_data: {genome, start_coordinate, end_coordinate, sequence}
  genome, start_coordinate, end_coordinate → sequence

Gibbs Domain Relations & Functional Dependencies

- gibbs_run: {output, gibbs_run_id, command, input, background_composition, priors, seed, version, frequency_map, near_optimal_map, maximal_map}
  gibbs_run_id → output, command, input, background_composition, priors, seed, version, frequency_map, near_optimal_map, maximal_map
- position: {position_id, gap, a_composition, t_composition, c_composition, g_composition, motif_id, position_number}
  position_id → gap, a_composition, t_composition, c_composition, g_composition, motif_id, position_number
  position_id ←← motif_id
- motif: {motif_id, gibbs_run_id}
  motif_id → gibbs_run_id
- binding_site_set: {binding_site_set_id, motif_id, gibbs_run_id}
  binding_site_set_id → motif_id, gibbs_run_id
- binding_site: {binding_site_set_id, binding_site_id, genome, start_coordinate, end_coordinate}
  binding_site_id → binding_site_set_id, genome, start_coordinate, end_coordinate

BMC Domain Relations and Functional Dependencies

- bmc_run: {bmc_run_id, 50_p_iters, command, divergent_genes, input, output, seed, version}
  bmc_run_id → 50_p_iters, command, divergent_genes, input, output, seed, version

- bmc_motif: {bmc_motif_id, bmc_cluster_id, p_motif, palindrome, present_iterations, run_id, regulon_id}
  bmc_motif_id → bmc_cluster_id, p_motif, palindrome, present_iterations, run_id, regulon_id

- bmc_cluster: {bmc_cluster_id, average_bayes_ratio, cluster_palindrome, cluster_present_iterations, fragmentation, proportion_size_average}
  bmc_cluster_id → average_bayes_ratio, cluster_palindrome, cluster_present_iterations, fragmentation, proportion_size_average

6.2.4 Logical Data Model (Database Schema)

The database schema implements the functional dependencies and facilitates the application perspectives. Entity-relationship diagrams for the schema are provided in appendix B. Figure B.1 shows the complete schema. Figure B.2 is the GenomeSharedData perspective, figure B.3 is the Gibbs perspective, and figure B.4 is the BMC perspective.

6.2.5 The Database Management System

PostgreSQL was selected for the DBMS because it appears to be the most capable, free, open source DBMS. In particular, PostgreSQL has strong support for relational integrity constraints, transaction processing, and stored procedures, as required for this system. Software libraries enabling access to a PostgreSQL database exist for a multitude of programming languages, including C, C++, Java, Perl, and PHP.

6.2.6 The User Interface

A prototype user interface has been implemented for the World Wide Web. This interface enables simple, platform-independent access to the database. The prototype has been implemented using PHP and the Apache web server. Later implementations

may leverage Java 2 Enterprise Edition (J2EE) technology such as Java Server Pages (JSPs) and Servlets to provide more advanced functionality.

The prototype interface is intended to illustrate only a subset of the benefits of this database system. Specifically, it allows basic queries to be executed. These queries should answer statistical questions about the biology in the data. For example, a biologist might want to see all Gibbs Sampler runs which predicted transcription factor binding sites for a specific gene in various genomes. Previously, queries such as this are difficult to answer using our collection of loosely connected data files, but they are significantly easier with this database system. The following queries were developed (with input from the users) to provide sufficient evidence of basic functionality:

1. Show all intergenic sequences from multiple species for a specific gene.
2. Show all Gibbs Sampler reports with binding sites predicted for a specific gene.
3. Show all Gibbs Sampler reports with predicted binding sites in a certain species with a minimum MAP value (maximum *a posteriori* probability).
4. Show all Gibbs Sampler reports with predicted binding sites for a particular gene with a minimum MAP value.

6.2.7 Application Software and Tools

To leverage our existing data and analysis reports, Perl scripts import existing data into the database. The DNA sequences of interest for the Cyanobacteria project are stored in approximately 1600 FASTA format files. These files are parsed, and appropriate data is inserted into the GenomeSharedData perspective of the database. Approximately 68,000 Gibbs Sampler reports exist, and these are mined for relevant information to insert into the Gibbs perspective of the database. Only a few BMC reports exist for this project, and they can be appropriately mined so their data can be inserted into the BMC perspective of the database.

6.3 System Evaluation

6.3.1 Web Interface Prototype

To evaluate the database system, a prototype web interface was developed to interface with the system. The prototype executes queries on the GenomeSharedData and

Gibbs perspectives. Appendix B.2 illustrates executing the example queries from section 6.2.6. All queries begin at the query page, shown in figure B.5.

Query 1 *Show all intergenic sequences from all species for gene groEL.*

To execute this query, from the query page, the user selects groEL from the topmost dropdown box, selects the "Get intergenic sequence data" radio button, and clicks the submit button. The results appear in both tabular and FASTA format for convenience. Please see figure B.6.

Query 2 *Show all Gibbs Sampler reports with binding sites predicted for gene purA.*

To execute this query, from the query page, the user selects purA from the topmost dropdown box, selects the "Search Gibbs Sampler results" radio button, and clicks the submit button. The results appear, as shown in figure B.7. From a list of Gibbs Sampler reports, the user can choose to view the details of a specific report by clicking the "View" link in the rightmost column. The details of the report appear, as shown in figure B.8. From this page, there are additional "View" links at the bottom associated with details about the motifs (mathematical models) and predicted transcription factor binding sites. The details on motifs and binding site predictions appear as shown in figures B.9 and B.10, respectively.

Query 3 *Show all Gibbs Sampler reports with predicted binding sites in species TELO with a minimum MAP value of 40.*

To execute this query, from the query page, the user selects TELO in the first dropdown box of the "Advanced Search" section, enters 40 for the minimum MAP value, and clicks submit. The results appear similar to figure B.7, and the user may continue to view further details from there.

Query 4 *Show all Gibbs Sampler reports with predicted binding sites for gene AT103 with a minimum MAP value of 20.*

To execute this query, from the query page, the user selects AT103 in the second dropdown box of the "Advanced Search" section, enters 40 for the minimum MAP value, and clicks submit. Again, the results appear similar to figure B.7, and the user may continue to view further details from there.

6.4 Technical Documentation

6.4.1 The Tools

The current tools consist of three Perl and two Bourne Shell scripts to import data to the database:

breakFa.pl This Perl script takes FASTA format sequence files and removes line breaks from the sequences. This could have been included in the main sequence import script, but this is a generally useful utility. The utility is named "breakFa.pl" because it breaks the FASTA (a.k.a. FA) format by removing line breaks which are supposed to be present after every 60 characters of sequence data.

parseIntergenics.pl This Perl script takes intergenic sequence files preprocessed by breakFa.pl and creates SQL scripts which import the sequences into the GenomeSharedData perspective of the database.

parseGibbs.pl This Perl script takes Gibbs Recursive Sampler report files and creates SQL scripts which import the appropriate data into the Gibbs perspective of the database.

doInsert.sh This Bourne Shell script cycles through a set of intergenic files, runs them through parseIntergenics.pl, and runs the resulting SQL scripts to insert the sequences into the database.

dbImport.sh This Bourne Shell script cycles through a set of compressed Gibbs report files, decompresses them, runs them through parseGibbs.pl, and runs the resulting SQL scripts to import the reports into the database.

Additional scripts still need to be written to parse BMC reports and insert appropriate data into the BMC perspective of the database.

6.4.2 The Web Interface Prototype

The web interface prototype consists of seven PHP pages. In their default configuration, they assume that the PostgreSQL server is running on the same machine as the Apache2/PHP4 web server. The connection options should be adjusted accordingly if this is not the case. The connection options should also be adjusted for the correct username and password to access the database. The current settings are appropriate for the server used to test the prototype. The role of each PHP page follows:

start.php This is the main query page. All users should enter the web interface here. For the basic queries (queries 1 and 2 above), this page calls getOrthologInfo.php. For the advanced queries (queries 3 and 4 above), this page calls getOrthologInfo2.php and getOrthologInfo3.php.

getOrthologInfo.php This page processes the basic queries and displays search results as demonstrated in figures B.6 and B.7. In the case of a Gibbs report search, the output gives the user the option to proceed to gibbsReport.php to retrieve the details of a Gibbs report.

getOrthologInfo2.php This page processes advanced queries on species and MAP value (query 3). Its output is similar to that of getOrthologInfo.php.

getOrthologInfo3.php This page processes advanced queries on gene and MAP value (query 4). Its output is similar to that of getOrthologInfo.php.

gibbsReport.php This page queries for the details on a specific Gibbs report and displays them to the user, as demonstrated in figure B.8. It allows the user to call motif.php or bindingSiteSet.php to retrieve details on motifs or sets of binding sites associated with that report.

motif.php This page displays the details of a motif associated with a Gibbs report, as shown in figure B.9. The numbers are the parameters used by the Gibbs Recursive Sampler for its product multinomial model of a motif.

bindingSiteSet.php This page displays the details about binding sites associated with a motif from a Gibbs report, as shown in figure B.10.

6.5 Summary

This database design has been implemented and tested with the PostgreSQL relational database management system (The PostgreSQL Global Development Group, 2004). The database schema is portable to any relational database system supporting SQL and foreign key relational integrity constraints. Data from an unpublished Cyanobacteria analysis project (McCue *et al.*, 2006) was imported to test the system, and a simple HTML/PHP (World Wide Web Consortium (W3C), 1997; The PHP Group, 2004) interface was also developed on an Apache web server (Foundation, 2004). PHP was chosen because it allowed for rapid development and testing. A

more robust platform, such as the Java 2 Enterprise Edition (J2EE), may be appropriate for a more refined web interface.

Chapter 7

OrthoGibbs

7.1 The OrthoGibbs Application

The OrthoGibbs application is the first application developed under the BRASS framework. OrthoGibbs implements a Gibbs Recursive Sampler with the Product Phylogeny model for *cis*-regulatory elements. Most of this functionality is designed as classes within the BRASS library, allowing the actual OrthoGibbs code to contain only a user interface which connects data to the processing logic contained in the library.

OrthoGibbs is also the next generation of the Gibbs Recursive Sampler (Thompson *et al.*, 2003). The BRASS framework provides a significantly cleaner code base and allows modular expansion of functionality in the future.

7.1.1 Input Parameters

During initialization, OrthoGibbs reads an XML (The World Wide Web Consortium, 2006) input file to determine its sampling behavior. An example input file is shown in figure 7.1. The example illustrates a typical run-time configuration. Parameters are either XML tags or attributes of XML tags. An XML tag defines a section of parameters, and its attributes provide additional details. Some tags also have nested tags, allowing additional flexibility to specify parameters. A full list of parameters is provided below.

7.1.1.1 The orthogibbs Tag

The `orthogibbs` tag is the root XML tag for an OrthoGibbs input file. The attributes of this tag are: `seed`, `numseeds`, `maxsites`, `sitesprior`, `iterations`, and `burnin`.

```
<?xml version="1.0"?>
<orthogibbs seed="42" maxsites="2" iterations="10000">
  <!-- declare sequence inputs with trees -->
  <sequencedata alphabet="DNA" background="composition" fulltree="lexA.6.tree">
    <sequence file="lexA.6.1.fa" format="FASTA" tree="lexA.6.1.tree" />
    <sequence file="lexA.6.2.fa" format="FASTA" tree="lexA.6.2.tree" />
    <sequence file="lexA.6.3.fa" format="FASTA" tree="lexA.6.3.tree" />
  </sequencedata>
  <!-- declare models -->
  <!-- this declares a default motif for OrthoGibbs -->
  <motif width="16" columns="phylogeny" pseudocounts="0.28" palindrome="on">
    <substitutionprocess process="F81" />
  </motif>
</orthogibbs>
```

Figure 7.1: Example OrthoGibbs Input

All attributes are optional. OrthoGibbs selects reasonable, but often suboptimal, values if none are specified.

The `seed` attribute allows the user to specify a specific random number seed. By running the program again with the same seed, the user should receive the same results. The default is the computer system time.

The `numseeds` attribute allows the user to specify the number of times the sampler should restart with a new seed for its random number generator. This allows the sampler to escape unfavorable results which may be the result of an unfortunate single seed. The default is 1.

The `maxsites` attribute allows the user to specify the maximum number of *cis*-regulatory elements per sequence. This corresponds to the maximum recursion depth in section 4.3.2. The default is 1.

The `sitesprior` attribute allows the user to specify a comma separated list of probability densities for the prior probability of $0, 1, 2, ..., n$ sites per sequence, where n is maxsites. The default is $1/(n+1)$.

The `iterations` attribute allows the user to specify the number of sampling iterations per seed. One iteration is a full cycle across all sequences under consideration. The default is 1000.

The `burnin` attribute allows the user to specify the number of iterations from the start of sampling to ignore for each seed. Early samples often contribute a high level of noise to the sampling process. The default is 20% of the total number of iterations.

7.1.1.2 The sequencedata Tag

The `sequencedata` tag provides information about the input sequences to OrthoGibbs. The attributes of this tag are: `alphabet`, `background`, and `fulltree`. All attributes

except `fulltree` are optional.

The `alphabet` attribute allows the user to specify the sequence alphabet, such as DNA, RNA, or protein. The default is DNA.

The `background` attribute allows the user to specify `composition` or `unified` for the background model. The `composition` setting uses the background composition as calculated directly from letter frequencies in the sequence data. The `unified` setting uses position specific background model information from the output of the Unified program (Liu & Lawrence, 1999). The default is composition.

The `fulltree` attribute allows the user to specify a file containing the phylogenetic tree (in Newick format (Joseph Felsenstein, 1986)) of all sequences in the analysis. This tree is used to compute sequence weights as described in section 4.4.1.

7.1.1.3 The sequence Tag

The `sequence` tag provides information about one clade of input sequences. A clade is a set of globally aligned sequences or an individual, unaligned sequence. The attributes of this tag are: `file`, `format`, `tree`, and `background`. `file` and `tree` are mandatory attributes, `format` is an optional attribute, and `background` is mandatory when Unified background model information will be used. More than one `sequence` tag should be provided as nested tags to the `sequencedata` tag.

The `file` attribute allows the user to specify a file containing the sequence data for the clade.

The `format` attribute allows the user to specify the file format for the file containing the sequence data for the clade. Currently, only `FASTA` is supported. The default is `FASTA`.

The `tree` attribute specifies the phylogenetic tree (in Newick format (Joseph Felsenstein, 1986)) of the clade. This tree is used in the computation of Felsenstein's Algorithm.

The `background` attribute allows the user to specify a file containing Unified background model information from the Unified program (Liu & Lawrence, 1999). This attribute is ignored if sequence composition is being used for the background model, but it is required if Unified output is being used for the background model.

7.1.1.4 The motif Tag

The `motif` tag provides information about a motif model for a *cis*-regulatory element. The attributes of this tag are: `width`, `columns`, `pseudocounts`, and `palindrome`.

`width` and `columns` are mandatory, but `pseudocounts` and `palindrome` are optional. More than one `motif` tag may be provided to instruct the sampler to sample multiple models.

The `width` attribute allows the user to specify the width of the model. The width is the number of adjacent letters of sequence which represent a *cis*-regulatory element.

The `columns` attribute allows the user to specify the type of model for a particular position in a *cis*-regulatory element. Currently, only `phylogeny` is supported, indicating the product phylogeny model.

The `pseudocounts` attribute allows the user to specify a default Bayesian pseudocount which is used to account for the prior probability of an observation. The default is 1.

The `palindrome` parameter allows the user to specify that a palindromic model is to be used. When set to `on`, this implies that the second half of the model is the reverse complement of the first half of the model. This constrains the model and reduces its degrees of freedom. The default is a non-palindromic model.

7.1.1.5 The substitutionprocess Tag

The `substitutionprocess` tag provides information about the substitution process used by Felsenstein's Algorithm. This tag has one mandatory attribute: `process`. Currently, only `F81` is valid, indicating the Felsenstein 1981 substitution process. This tag is always a nested tag of `motif`.

7.2 Evaluating Solutions

OrthoGibbs reports a solution after it has run for the specified number of iterations. A solution is a set of potential *cis*-regulatory elements in the sequences under consideration. The solution reported is called a "50% frequency solution". After the specified number of burn-in iterations, the sampler counts each time a *cis*-regulatory element is sampled in any subsequent iteration. Any *cis*-regulatory element which is sampled in at least 50% of the iterations after the burn-in period is reported in the solution. This type of solution may also be called a "centroid" solution.

The centroid approach to evaluating solutions has two distinct advantages over a maximum *a posteriori* (MAP) solution. First, counting samples is computationally much simpler than calculating a probability. Second, the solution contains a representative sample of the most likely solutions within the sampling space. This approach works extremely well for RNA secondary structure sampling, and it is likely to be

much more widely applicable (Ding *et al.*, 2005). Intuitively, the centroid approach accounts for some uncertainty about the molecular biology behind the mathematical model. A maximum *a posteriori* or optimal solution is a single, likely solution under a particular model. A centroid solution is a set of likely solutions under the model. The true, biological solution is more likely to be represented in such a set than by a single solution.

7.3 Testing OrthoGibbs on Synthetic Data

7.3.1 Generating the Data

Five data sets were created to simulate real data which could be processed by OrthoGibbs. Each data set consisted of 100 simulated intergenic regions containing 500 nucleotides upstream of a hypothetical gene from eight related species. The eight species were chosen to simulate *E. coli* and its close relatives in the gamma proteobacteria family:

1. *Escherichia coli*
2. *Shigella flexneri*
3. *Salmonella enterica Typhi*
4. *Salmonella bongori*
5. *Citrobacter rodentium*
6. *Klebsiella pneumoniae*
7. *Proteus mirabilis*
8. *Vibrio cholera el Tor*

These species are related by the tree shown in figure 7.2. The nucleotides were drawn from the Felsenstein 1981 model of phylogeny (Felsenstein, 1981) using this tree.

The first synthetic data set is a negative control containing no simulated *cis*-regulatory elements. The other four data sets contain one, two, three, or four planted *cis*-regulatory elements, respectively. The planted *cis*-regulatory elements were also drawn from the Felsenstein 1981 model using a 22 nucleotide position weight matrix of cyclic AMP receptor protein (CRP) as the equilibrium distribution. CRP is a well characterized *cis*-regulatory element present in all of these species. Its position weight matrix is similar to the one shown in figure 1.1.

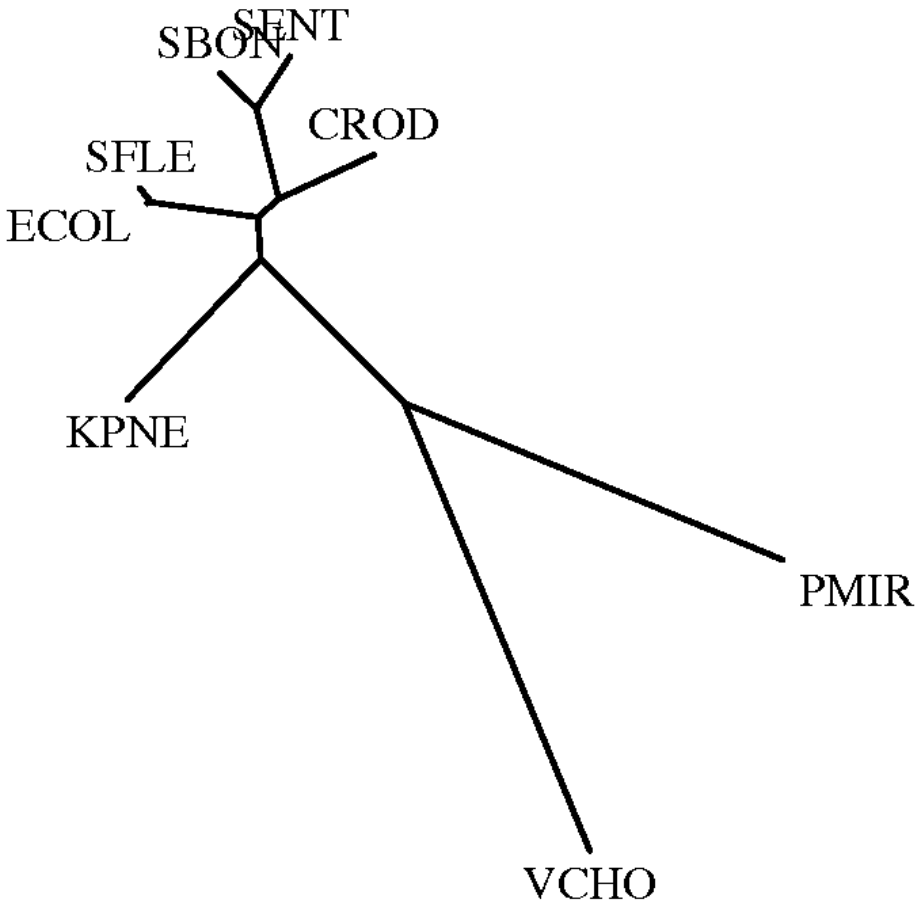


Figure 7.2: Phylogenetic Tree of Species in Test Data
ECOL: *E. coli*; SFLE: *S. flexneri*; SENT: *S. enterica Typhi*; SBON: *S. bongori*; CROD: *C. rodentium*; KPNE: *K. pneumoniae*; PMIR: *P. mirabilis*; VCHO: *V. cholera el Tor*

7.3.2 Running the Experiment

The parameters to OrthoGibbs are important for both obtaining and interpreting results. The most commonly adjusted parameters are the pseudocounts for each position in the model (motif) of the *cis*-regulatory element, the maximum number of *cis*-regulatory elements per sequence, and the prior probabilities for the number of *cis*-regulatory elements per sequence. For this experiment, uniform pseudocounts of 0.28 were used for each position in the motif. These pseudocounts are used when sampling an equilibrium distribution for Felsenstein's Algorithm, and they provide an expected value with one bit of information content. We also used a palindromic motif in this experiment because many transcription factors, including CRP, have palindromic *cis*-regulatory elements. We allowed up to two *cis*-regulatory elements per sequence, and we provided uniform prior probabilities for 0, 1, or 2 sites per sequence. We also assumed that the simulated sequences from the last two species were independent and not globally alignable because this would be true if we were using biological data.

7.3.3 Results

We measure our results in *E. coli*, our species of interest; true and false positives (hits or misses against planted *cis*-regulatory elements) are only counted for predictions in the global alignment containing the simulated *E. coli* sequence.

Generally, we are interested in three measures of performance: sensitivity, specificity, and positive predictive value. Sensitivity is the percentage of actual (or in this case, planted) *cis*-regulatory elements correctly predicted by OrthoGibbs in the positive control data. Specificity is the percentage of negative control data which is correctly characterized as containing no *cis*-regulatory elements by OrthoGibbs.

To evaluate OrthoGibbs's performance on the negative control data, we want to count the total number of predictions and the number of intergenics with at least one prediction. Every prediction in this data set is a false positive. Out of the 100 intergenics in the experiment, OrthoGibbs predicted only a single site in three intergenics. This corresponds to specificity (by *cis*-regulatory element and by intergenic) of 97%.

To evaluate OrthoGibbs's performance on the positive control data, we want to answer the following questions:

1. How many predictions by OrthoGibbs overlap one of the $100N$ planted sites? (true positive by site)

	Data Set 1	Data Set 2	Data Set 3	Data Set 4
Question 1	17	116	154	176
Question 2	83	84	146	224
Question 3	5	2	0	0
Question 4	17	61	82	93
Question 5	83	45	100*	100*
Question 6	4	2	0	0
Sensitivity	17%	58%	51%	44%
PPV	77%	98%	100%	100%

Table 7.1: OrthoGibbs Results on Synthetic Positive Control Data
* Since a maximum of two sites per sequence were allowed, OrthoGibbs had to miss at least one true site.

2. How many of the $100N$ sites are not predicted by OrthoGibbs? (false negative by site)

3. How many predictions by OrthoGibbs do not overlap one of the $100N$ sites? (false positive by site)

4. In how many of the 100 intergenics does OrthoGibbs find at least one true site? (true positive by intergenic)

5. In how many of the 100 intergenics does OrthoGibbs miss at least one true site? (false negative by intergenic)

6. In how many of the 100 intergenics does OrthoGibbs make at least one false prediction? (false positive by intergenic)

For a prediction to overlap a planted site, we specify that it must overlap at least half of the site. From these questions, we can calculate the sensitivity and positive predictive value. The results for the four positive control data sets are shown in table 7.1. The data set number N corresponds to having $100N$ planted sites.

7.4 Testing OrthoGibbs on Real Data

7.4.1 Selecting the Data

A biological data set was selected from the gamma proteobacteria genomes listed in section 7.3.1. The following criteria were used to select sequences for the data set:

1. The sequences must be intergenic regions upstream of orthologous genes which are present in all eight species.

2. The *E. coli* sequence must contain an experimentally verified *cis*-regulatory element.

3. The orthologous sequences from the first six species must be globally alignable with CLUSTALW (Chenna *et al.*, 2003) using its default parameters.

4. The intergenic sequence length must be 500 nucleotides or less. In bacterial data, this is the typical region where *cis*-regulatory elements are expected.

These selection criteria yielded a data set containing 72 intergenic sequences.

7.4.2 Running the Experiment

The biological data set was processed under the same set of conditions described in section 7.3.2. This is not ideal, but it is a reasonable way to analyze such a data set without tuning the parameters for specific transcription factors or intergenic sequences. Many cis-regulatory elements in the gamma proteobacteria are palindromic and less than 22 nucleotides long.

7.4.3 Results

The same statistics reported for the synthetic data were calculated and are reported in table 7.2. Interpreting results for this data set is quite difficult. From the data set defined in section 7.4.1, we have 109 detectable *cis*-regulatory elements. These elements are deemed detectable because the experimentally verified *cis*-regulatory element can be discovered in the *E. coli* sequence with a simple string search. The distribution of detectable sites per intergenic is shown in figure 7.3.

Some of the 109 detectable *cis*-regulatory elements may not actually be detectable to OrthoGibbs for various reasons. For example, the *cis*-regulatory element in *E. coli* may have a gap in one or more of its orthologous counterparts introduced by the global alignment process. OrthoGibbs cannot predict *cis*-regulatory elements in regions of globally aligned sequence with gaps in any of the constituent sequences.

Since OrthoGibbs was run with a maximum of two sites per sequence, OrthoGibbs cannot predict all of the sites in sequences which contain more than two detectable

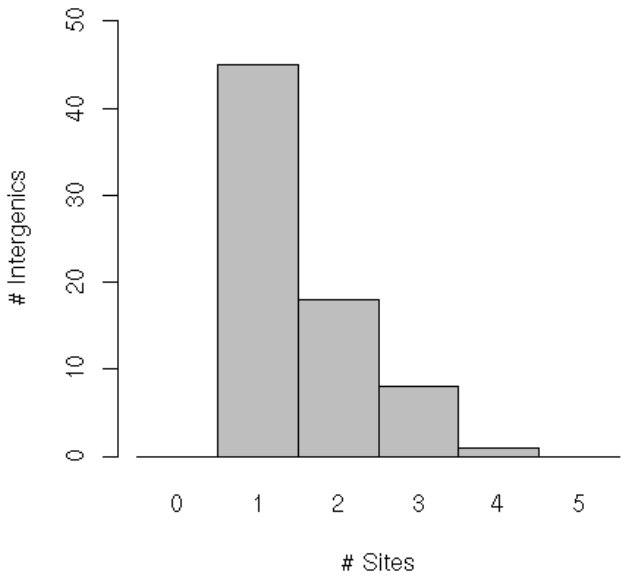


Figure 7.3: Detectable Sites per Intergenic

sites. Of the 72 intergenic sequences in the data set, 9 contain more than two detectable sites. Of the 109 detectable sites, 10 are not actually detectable to OrthoGibbs when run with a maximum of two sites per sequence.

Biological anomalies may also influence the detection of some *cis*-regulatory elements. For example, two *cis*-regulatory elements upstream of the argR gene are directly adjacent. One OrthoGibbs prediction can overlap two *cis*-regulatory elements.

Using heuristics, we were able to tune the prior probabilities on the number of *cis*-regulatory elements per sequence to produce better results than those obtained using uniform priors. These results are also shown in table 7.2.

	Biological Data Set	Biological Data Set, Tuned Priors
Question 1	25	28
Question 2	84	81
Question 3	26	21
Question 4	19	20
Question 5	60	60
Question 6	21	16
Sensitivity	23%	26%
PPV*	50%	57%

Table 7.2: OrthoGibbs Results on Biological Data
* Since this is biological data, PPV may be pessimistic because a false prediction may be an unreported site.

Chapter 8

Discussion and Conclusions

8.1 Comparison of OrthoGibbs to Other Methods

Table 8.1 provides a brief overview of the similarities and differences between OrthoGibbs and its peer algorithms described in section 2.3. Direct comparisons between results produced by each application are extremely difficult to perform. This is primarily due to the parameters which may be adjusted for each algorithm and/or data set.

Benchmarking of similar algorithms has been performed by a large group of authors, and their discussion regarding the difficulties of benchmarking these algorithms is quite insightful (Tompa *et al.*, 2005). They encountered the same difficulties discussed in chapter 7 with respect to generating and/or selecting data sets, tuning parameters, and interpreting results.

8.2 Benefits of BRASS

One of the greatest problems in our field is the integration of data from multiple sources (Chicurel, 2002). Bioinformatics data must be freely available in a machine-readable format to foster collaboration (Stein, 2002) and ensure scientific integrity. Many bioinformatics labs provide only a web interface to their data, and this makes it difficult for other labs to verify their results or incorporate them into a new analysis. The software framework will include an open data model accessible via standard transport mechanisms (Dowell *et al.*, 2001; Wilkinson & Links, 2002) to facilitate these tasks.

Algorithm	Type?	Tree?	Seq. Indep?
PhyloCon	Optimization	no	yes
OrthoMEME	EM	no	no
EMnEM	EM	yes	no
PhyME	EM	yes	no
CompareProspector	Gibbs	no	yes
Wong	Gibbs	yes	no
PhyloGibbs	Gibbs	yes**	no
OrthoGibbs	Gibbs	yes	no
	Pre-align?	> 2 Species?	> 1Motif?
PhyloCon	yes	yes	yes
OrthoMEME	no	no	no
EMnEM	yes	yes	no
PhyME	yes	yes	no
CompareProspector	yes	yes	yes
Wong	no	yes	no
PhyloGibbs	yes*	yes	yes
OrthoGibbs	yes*	yes	yes

Table 8.1: Comparison of Algorithms

* unaligned data may also be used, and it is treated as phylogenetically independent

** officially supports star topology only

Description of columns: (1) What type of algorithm is used? (2) Is a phylogenetic tree used? (3) Are the sequences considered statistically independent? (4) Must the orthologous sequences be aligned? (5) Can sequences from more than two species be considered? (6) Can more than one motif be discovered?

8.2.1 Software Framework Benefits

The software framework provides an object representation for the Gibbs Phylogenetic Sampler and generically lays the groundwork to accommodate other statistical bioinformatics applications. Many sampling strategies, biological data structures, and mathematical models may be shared between various analysis applications. For example, the Bayesian Motif Clustering (BMC) application (Qin *et al.*, 2003) and the *SCAN* application (Neuwald *et al.*, 1995) can use product multinomial motifs from the Gibbs Recursive Sampler as input. If future versions of these applications use the framework, more seamless integration will be possible. Instead of passing intermediate files between applications, the data objects can directly be shared. As an example, a new class hierarchy may provide some of the software glue to process a workflow leading from Gibbs to either of these applications.

One advantage of abstractly defining our methods in a software framework is that it will be easier to develop new applications that use components or patterns from other applications. For example, a software developer could write a composite application that encompasses the functionality of several framework applications with a shared user interface. It would also be possible to reuse specific pieces of an application within another, such as a sampling procedure.

Another advantage of the framework is that it will ease the transition to grid computing. Many statistical bioinformatics applications can take advantage of distributed or parallel processing. Implementing distributed or parallel applications is often a challenge because the computational process scheduling can become intertwined with the application specific data processing. With the framework's object-oriented design, it will be easier to separate the process scheduling logic from the application logic. For example, it is typical to run the same application with many different random number generator seeds and take the best result. A grid-aware application might accomplish this by instantiating and executing several sampler objects. It might even be possible for the sampler objects in this application to share constant data directly depending on the grid's memory structure. Developing an application in this manner is much simpler than rewriting an entire application to be aware of all of these details.

8.2.2 Database System Benefits

The database system provides a standard representation of our most important data structures, means of integrating information from various sources, a central repository for long term storage, and accessibility which conveniently facilitates the creation of

web interfaces.

With a standard representation of data structures, new applications developed for sequence analysis can have an authoritative, language independent reference. The data model can be represented in UML, and a competent software developer can design matching structures in any programming language. For example, as mentioned earlier, the BMC application uses some of the output of the Gibbs Sampler as input. Currently, the process of bringing data from the Gibbs Sampler to BMC involves several Perl scripts parsing many text files. Also, the formats of those text files may change from time to time, and the Perl scripts have to be modified. With the data stored in the database, BMC needs to know only the structures in the database, which are less likely to change (though they may be transparently expanded).

With an integrated source for all of our data, queries which were previously very impractical can be greatly simplified. Currently, some queries involve writing Perl scripts to parse multitudes of files. With the database, some of these queries will require writing only some SQL code. Even in the worst case of a very complex SQL query, the user will not have to worry about parsing many files of multiple formats to integrate the necessary information.

With a central repository for long term storage, our data will be more easily accessible to scientists both inside and outside my lab. Currently, results that may interest multiple people in my lab are scattered across several file systems on a shared file server. Instead of searching various users' home directories and project directories, we will only need to search through our database schema.

When we want to make results public, we often take some sort of static snapshot and make it viewable on the web. It is not always clear where the data in the web version originated, and this makes it much less useful. It can also be quite difficult to track our process backwards to find the origin. By creating a web interface to the database, we can directly tie our front end interface to our back end processing. We can know where the data originates because it is being pulled directly from the database.

References

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. & Walter, P. (2002) *Molecular Biology of the Cell.* 4th edition„ Garland Science.

Bailey, T. L. & Elkan, C. P. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, (Altman, R., Brutlag, D., Karp, P., Lathrop, R. & Searls, D., eds), pp. 28–36 American Association for Artificial Intelligence AAAI Press, Stanford University, Palo Alto, CA.

Benos, P., Lapedes, A. & Stormo, G. (2002) Is there a code for protein-DNA recognition? Probab(ilistical)ly... *Bioessays,* **24** (5), 466–75.

Blanchette, M., Kent, W. J., Riemer, C., Elnitski, L., Smit, A. F., Roskin, K. M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E. D., Haussler, D. & Miller, W. (2004) Aligning Multiple Genomic Sequences With the Threaded Blockset Aligner. *Genome Res.,* **14** (4), 708–715.

Boost.org (2004). C++ Boost Libraries. http://www.boost.org/.

Brudno, M., Do, C. B., Cooper, G. M., Kim, M. F., Davydov, E., Program, N. C. S., Green, E. D., Sidow, A. & Batzoglou, S. (2003) LAGAN and Multi-LAGAN: Efficient Tools for Large-Scale Multiple Alignment of Genomic DNA. *Genome Res.,* **13** (4), 721–731.

Chenna, R., Sugawara, H., Koike, T., Lopez, R., Gibson, T. J., Higgins, D. G. & Thompson, J. D. (2003) Multiple sequence alignment with the Clustal series of programs. *Nucl. Acids Res.,* **31** (13), 3497–3500.

Chicurel, M. (2002) Bioinformatics: bringing it all together. *Nature,* **419** (6908), 751, 753, 755 passim.

Cole, J., Chai, B., Marsh, T., Farris, R., Wang, Q., Kulam, S., Chandra, S., McGarrell, D., Schmidt, T., Garrity, G. & Tiedje, J. (2003) The Ribosomal Database Project (RDP-II): previewing a new autoaligner that allows regular updates and the new prokaryotic taxonomy. *Nucleic Acids Res,* **31** (1), 442–3.

Ding, Y., Chan, C. Y. & Lawrence, C. E. (2005) RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA,* **11** (8), 1157–1166.

Ding, Y. & Lawrence, C. (2003) A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Res,* **31** (24), 7280–301.

Dowell, R., Jokerst, R., Day, A., Eddy, S. & Stein, L. (2001) The distributed annotation system. *BMC Bioinformatics,* **2** (1), 7.

Fayad, M. E., Schmidt, D. C. & Johnson, R. E., eds (1999*a*) *Building Application Frameworks: Object-Oriented Foundations of Framework Design*. John Wiley & Sons.

Fayad, M. E., Schmidt, D. C. & Johnson, R. E., eds (1999*b*) *Implementing Application Frameworks: Object-Oriented Frameworks at Work*. John Wiley & Sons.

Felsenstein, J. (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. *J Mol Evol,* **17** (6), 368–376. PubMed 7288891.

Felsenstein, J. (1993) *PHYLIP (Phylogeny Interface Package) 3.5c*. Department of Genetics, University of Washington, Seattle, WA.

Felsenstein, J. (2001) Taking variation of evolutionary rates between sites into account in inferring phylogenies. *J Mol Evol,* **53** (4–5), 447–455. PubMed 11675604.

Felsenstein, J. (2003) *Inferring Phylogenies*. Sinauer Associates.

Felsenstein, J. & Churchill, G. A. (1996) A hidden Markov model approach to variation among sites in rate of evolution. *Mol Biol Evol,* **13** (1), 93–104. PubMed 8583911.

Foundation, T. A. S. (2004). Apache HTTP server 2.0. http://httpd.apache.org/.

Fox, G., Stackebrandt, E., Hespell, R., Gibson, J., Maniloff, J., Dyer, T., Wolfe, R., Balch, W., Tanner, R., Magrum, L., Zablen, L., Blakemore, R., Gupta, R., Bonen, L., Lewis, B., Stahl, D., Luehrsen, K., Chen, K. & Woese, C. (1980) The phylogeny of prokaryotes. *Science,* **209** (4455), 457–63.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Hasegawa, M., Kishino, H. & Yano, T. (1985) Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *J Mol Evol,* **22** (2), 160–174. PubMed 3934395.

Heidelberg, J., Paulsen, I., Nelson, K., Gaidos, E., Nelson, W., Read, T., Eisen, J., Seshadri, R., Ward, N., Methe, B., Clayton, R., Meyer, T., Tsapin, A., Scott, J., Beanan, M., Brinkac, L., Daugherty, S., DeBoy, R., Dodson, R., Durkin, A., Haft, D., Kolonay, J., Madupu, R., Peterson, J., Umayam, L., White, O., Wolf, A., Vamathevan, J., Weidman, J., Impraim, M., Lee, K., Berry, K., Lee, C., Mueller, J., Khouri, H., Gill, J., Utterback, T., McDonald, L., Feldblyum, T., Smith, H., Venter, J., Nealson, K. & Fraser, C. (2002) Genome sequence of the dissimilatory metal ion-reducing bacterium Shewanella oneidensis. *Nat Biotechnol,* **20** (11), 1118–23.

Hertz, G. & Stormo, G. (1999) Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics,* **15** (7-8), 563–77.

Hertz, G. Z. & Stormo, G. D. (1994) Identification of consensus patterns in unaligned DNA and protein sequences: a large-deviation statistical basis for penalizing gaps. In *Proceedings of the Third International Conference on Bioinformatics and Genome Research*, (Lim, H. A. & Cantor, C. R., eds), pp. 201–216 World Scientific Publishing Co., Singapore, Tallahassee, FL.

Hillis, D. M., Moritz, C. & Mable, B. K., eds (1996) *Molecular Systematics*. 2nd edition„ Sinauer Associates, Inc., Sunderland, MA.

Holmes, I. & Bruno, W. J. (2001) Evolutionary HMMs: a Bayesian approach to multiple alignment. *Bioinformatics,* **17** (9), 803–820.

Holt, J. G., ed. (1994) *Bergey's Manual of Determinative Bacteriology*. 9th edition„ Williams & Wilkins, Baltimore.

Jacob, F. & Monod, J. (1961) Genetic regulatory mechanisms in the synthesis of proteins. *J Mol Biol,* **3**, 318–56.

Jamison, D. (2003) Open bioinformatics. *Bioinformatics,* **19** (6), 679–80.

Johnson, R. E. (1997) Frameworks = (components + patterns). *Commun. ACM,* **40** (10), 39–42.

Joseph Felsenstein (1986). The Newick tree format. http://evolution.genetics.washington.edu/phylip/newicktree.html.

Jukes, T. H. & Cantor, C. (1969) Evolution of protein molecules. In *Mammalian Protein Metabolism*, (Munro, H. M., ed.), vol. 3,. Academic Press New York, NY pp. 21–132.

Kimura, M. (1980) A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J Mol Evol,* **16** (2), 111–120. PubMed 7463489.

Lanave, C., Preparata, G., Saccone, C. & Serio, G. (1984) A new method for calculating evolutionary substitution rates. *J Mol Evol,* **20** (1), 86–93. PubMed 6429346.

Lawrence, C., Altschul, S., Boguski, M., Liu, J., Neuwald, A. & Wootton, J. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science,* **262** (5131), 208–14.

Lawrence, C. & Reilly, A. (1990) An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins,* **7** (1), 41–51.

Lewin, B. (2000) *Genes VII.* Oxford University Press.

Li, X. & Wong, W. H. (2005) Sampling motifs on phylogenetic trees. *PNAS,* **102** (27), 9481–9486.

Liu, J. & Lawrence, C. (1999) Bayesian inference on biopolymer models. *Bioinformatics,* **15** (1), 38–52.

Liu, J., Neuwald, A. & Lawrence, C. (1995) Bayesian models for multiple local sequence alignment and Gibbs sampling strategies. *J. Amer Stat. Assoc.,* **90**, 1156–70.

Liu, J. S. (1994) The Collapsed Gibbs Sampler in Bayesian Computations With Applications to a Gene Regulation Problem. *Journal of the American Statistical Association,* **89** (427), 958–966.

Liu, J. S. (2001) *Monte Carlo Strategies in Scientific Computing.* Springer Series in Statistics, Springer-Verlag, New York, NY.

Liu, J. S., Neuwald, A. F. & Lawrence, C. E. (1999) Markovian Structures in Biological Sequence Alignments. *J. Amer. Statist. Assoc.,* **94**, 1–15.

Liu, X. S., Brutlag, D. L. & Liu, J. S. (2001) BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. In *Pacific Symposium on Biocomputing*, (Altman, R. B., Dunker, A. K., Hunker, L., Lauderdale, K. & Klein, T. E., eds), pp. 127–138 World Scientific Publishing Co., Singapore, The Orchid at Mauna Lani, HI.

Liu, Y., Liu, X. S., Wei, L., Altman, R. B. & Batzoglou, S. (2004) Eukaryotic Regulatory Element Conservation Analysis and Identification Using Comparative Genomics. *Genome Res.,* **14** (3), 451–458.

Lovley, D. & Lloyd, J. (2000) Microbes with a mettle for bioremediation. *Nat Biotechnol,* **18** (6), 600–1.

Mangalam, H. (2002) The Bio* toolkits–a brief overview. *Brief Bioinform,* **3** (3), 296–302.

Matthews, B. (1988) Protein-DNA interaction. No code for recognition. *Nature,* **335** (6188), 294–5.

McCue, L., Thompson, W., Carmack, C. & Lawrence, C. (2002) Factors influencing the identification of transcription factor binding sites by cross-species comparison. *Genome Res,* **12** (10), 1523–32.

McCue, L., Thompson, W., Carmack, C., Ryan, M., Liu, J., Derbyshire, V. & Lawrence, C. (2001) Phylogenetic footprinting of transcription factor binding sites in proteobacterial genomes. *Nucleic Acids Res,* **29** (3), 774–82.

McCue, L. A., Smith, T. M., Thompson, W., Palumbo, M. & Lawrence, C. (2006). Whole genome phylogenetic footprinting and regulon prediction in cyanobacteria. In preparation.

Mitchison, G. J. (1999) A probabilistic treatment of phylogeny and sequence alignment. *J Mol Evol,* **49** (1), 11–22.

Mitchison, G. J. & Durbin, R. (1995) Tree-based maximal likelihood substitution matrices and hidden Markov models. *J Mol Evol,* **41**, 1139–1151.

Moses, A., Chiang, D. & Eisen, M. (2004) Phylogenetic motif detection by expectation-maximization on evolutionary mixtures. In *Ninth Pacific Symposium on Biocomputing (PSB)*.

Nealson, K. & Little, B. (1997) Breathing manganese and iron: solid-state respiration. *Advances in Applied Microbiology,* **45**, 213–39.

Neuwald, A., Liu, J. & Lawrence, C. (1995) Gibbs motif sampling: detection of bacterial outer membrane protein repeats. *Protein Sci,* **4** (8), 1618–32.

Newberg, L. A., McCue, L. A. & Lawrence, C. E. (2005) The Relative Inefficiency of Sequence Weights Approaches in Determining a Nucleotide Position Weight Matrix. *Statistical Applications in Genetics and Molecular Biology,* **4**, (reviewed).

Neyman, J. (1971) Molecular studies of evolution: a source of novel statistical problems. In *Statistical Decision Theory and Related Topics*, (Gupta, S. S. & Yackel, J., eds),. Academic Press New York, NY pp. 1–27.

Olsen, G., Woese, C. & Overbeek, R. (1994) The winds of (evolutionary) change: breathing new life into microbiology. *J Bacteriol,* **176** (1), 1–6.

Pabo, C. & Sauer, R. (1984) Protein-DNA recognition. *Annu Rev Biochem,* **53**, 293–321.

Prakash, A., Blanchette, M., Sinha, S. & Tompa, M. (2004) Motif discovery in heterogeneous sequence data. In *Ninth Pacific Symposium on Biocomputing (PSB)*.

Qin, Z., McCue, L., Thompson, W., Mayerhofer, L., Lawrence, C. & Liu, J. (2003) Identification of co-regulated genes through Bayesian clustering of predicted regulatory binding sites. *Nat Biotechnol,* **21** (4), 435–9.

Quackenbush, J. (2003) Open-source software accelerates bioinformatics. *Genome Biol,* **4** (9), 336.

Rice, P., Longden, I. & Bleasby, A. (2000) EMBOSS: the European Molecular Biology Open Software Suite. *Trends Genet,* **16** (6), 276–7.

Roth, F. P., Hughes, J. D., Estep, P. W. & Church, G. M. (1998) Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nat. Biotech.,* **16** (10), 939–945.

Schmidt, D. C., Gokhale, A. & Natarajan, B. (2004) Frameworks: Why They are Important and How to Apply Them Effectively. *ACM Queue magazine,* **2** (5), (advance publication on author's web page).

Schneider, T. & Stephens, R. (1990) Sequence Logos: A New Way to Display Consensus Sequences. *Nucl. Acids Res.,* **18**, 6097–6100.

Siddharthan, R., Siggia, E. D. & van Nimwegen, E. (2005) PhyloGibbs: A Gibbs Sampling Motif Finder That Incorporates Phylogeny. *PLoS Computational Biology,* **1** (7).

Siepel, A. & Haussler, D. (2004) Phylogenetic estimation of context-dependent substitution rates by maximum likelihood. *Mol Biol Evol,* **21** (3), 468–488. PubMed 14660683.

Smith, T. M. (2003). Computationally inferring the number of transcription factor binding motifs from DNA sequence data. Master's thesis, Rensselaer Polytechnic Institute. Adviser: Charles E. Lawrence.

Stein, L. (2002) Creating a bioinformatics nation. *Nature,* **417** (6885), 119–20.

Stormo, G. D. & Hartzell, 3rd, G. W. (1989) Identifying protein-binding sites from unaligned DNA fragments. *Proc Natl Acad Sci U S A,* **86** (4), 1183–1187.

Stroustrup, B. (2000) *The C++ Programming Language.* Special 3rd edition„ Addison-Wesley.

The Apache Software Foundation (2005). Xerces C++ Parser. http://xml.apache.org/xerces-c/.

The PHP Group (2004). PHP 4.3. http://www.php.net/.

The PostgreSQL Global Development Group (2004). PostgreSQL 7.4. http://www.postgresql.org/.

The World Wide Web Consortium (2006). Extensible Markup Language (XML). http://www.w3.org/XML/.

Thomas, J. W., Touchman, J. W., Blakesley, R. W., Bouffard, G. G., Beckstrom-Sternberg, S. M., Margulies, E. H., Blanchette, M., Siepel, A. C., Thomas, P. J., McDowell, J. C., Maskeri, B., Hansen, N. F., Schwartz, M. S., Weber, R. J., Kent, W. J., Karolchik, D., Bruen, T. C., Bevan, R., Cutler, D. J., Schwartz, S., Elnitski,

L., Idol, J. R., Prasad, A. B., Lee-Lin, S. Q., Maduro, V. V., Summers, T. J., Portnoy, M. E., Dietrich, N. L., Akhter, N., Ayele, K., Benjamin, B., Cariaga, K., Brinkley, C. P., Brooks, S. Y., Granite, S., Guan, X., Gupta, J., Haghighi, P., Ho, S. L., Huang, M. C., Karlins, E., Laric, P. L., Legaspi, R., Lim, M. J., Maduro, Q. L., Masiello, C. A., Mastrian, S. D., McCloskey, J. C., Pearson, R., Stantripop, S., Tiongson, E. E., Tran, J. T., Tsurgeon, C., Vogt, J. L., Walker, M. A., Wetherby, K. D., Wiggins, L. S., Young, A. C., Zhang, L. H., Osoegawa, K., Zhu, B., Zhao, B., Shu, C. L., De Jong, P. J., Lawrence, C. E., Smit, A. F., Chakravarti, A., Haussler, D., Green, P., Miller, W. & Green, E. D. (2003) Comparative analyses of multi-species sequences from targeted genomic regions. *Nature,* **424** (6950), 788–793. PubMed 12917688.

Thompson, W., Palumbo, M. J., Wasserman, W. W., Liu, J. & Lawrence, C. (2004) Decoding human regulatory circuits. *Genome Res.,* **14**, (reviewed).

Thompson, W., Rouchka, E. & Lawrence, C. (2003) Gibbs Recursive Sampler: finding transcription factor binding sites. *Nucleic Acids Res,* **31** (13), 3580–5.

Thorne, J. L., Kishino, H. & Felsenstein, J. (1991) An evolutionary model for maximum likelihood alignment of DNA sequences. *J Mol Evol,* **33** (2), 114–124.

Thorne, J. L., Kishino, H. & Felsenstein, J. (1992) Inching toward reality: an improved likelihood model of sequence evolution. *J Mol Evol,* **34** (1), 3–16.

Tompa, M., Li, N., Bailey, T. L., Church, G. M., De Moor, B., Eskin, E., Favorov, A. V., Frith, M. C., Fu, Y., Kent, W. J., Makeev, V. J., Mironov, A. A., Noble, W. S., Pavesi, G., Pesole, G., Regnier, M., Simonis, N., Sinha, S., Thijs, G., van Helden, J., Vandenbogaert, M., Weng, Z., Workman, C., Ye, C. & Zhu, Z. (2005) Assessing computational tools for the discovery of transcription factor binding sites. *Nat Biotechnol,* **23** (1), 137–144.

van Heesch, D. (2005). Doxygen. http://www.doxygen.org/.

Voet, D. & Voet, J. G. (2004) *Biochemistry*. 3rd edition„ Wiley Text Books.

Wang, T. & Stormo, G. D. (2003*a*) Combining phylogenetic data with co-regulated genes to identify regulatory motifs. *Bioinformatics,* **19** (18), 2369–2380.

Wang, T. & Stormo, G. D. (2003*b*) Combining phylogenetic data with co-regulated genes to identify regulatory motifs. *Bioinformatics,* **19** (18), 2369–2380.

Wilkinson, M. & Links, M. (2002) BioMOBY: an open source biological web services proposal. *Brief Bioinform,* **3** (4), 331–41.

Woese, C. & Fox, G. (1977) Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proc Natl Acad Sci U S A,* **74** (11), 5088–90.

Woese, C. R., Fox, G. E., Zablen, L., Uchida, T., Bonen, L., Pechman, K., Lewis, B. J. & Stahl, D. (1975) Conservation of Primary Structure in 16S rRNA. *Nature,* **254**, 83–85.

Workman, C. & Stormo, G. (2000) ANN-Spec: a method for discovering transcription factor binding sites with improved specificity. *Pac Symp Biocomput,* **5**, 467–78.

World Wide Web Consortium (W3C) (1997). HyperText markup language HTML 4.01. http://www.w3c.org/MarkUp/.

Yang, Z. (1993) Maximum likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites. *Mol. Biol. Evol.,* **10**, 1396–1401.

Yang, Z. (1994*a*) Estimating the pattern of nucleotide substitution. *J Mol Evol,* **39** (1), 105–111. PubMed 8064867.

Yang, Z. (1994*b*) Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *J Mol Evol,* **39** (3), 306–314. PubMed 7932792.

Yang, Z. (1995) A space-time process model for the evolution of DNA sequences. *Genetics,* **139** (2), 993–1005.

Zuckerkandl, E. & Pauling, L. (1965) Molecules as documents of evolutionary history. *J Theor Biol,* **8** (2), 357–66.

Appendix A

BRASS Supplemental Materials

A.1 BRASS UML Diagrams

The UML diagrams on the following pages illustrate the structures and interfaces of all major components in the BRASS library.

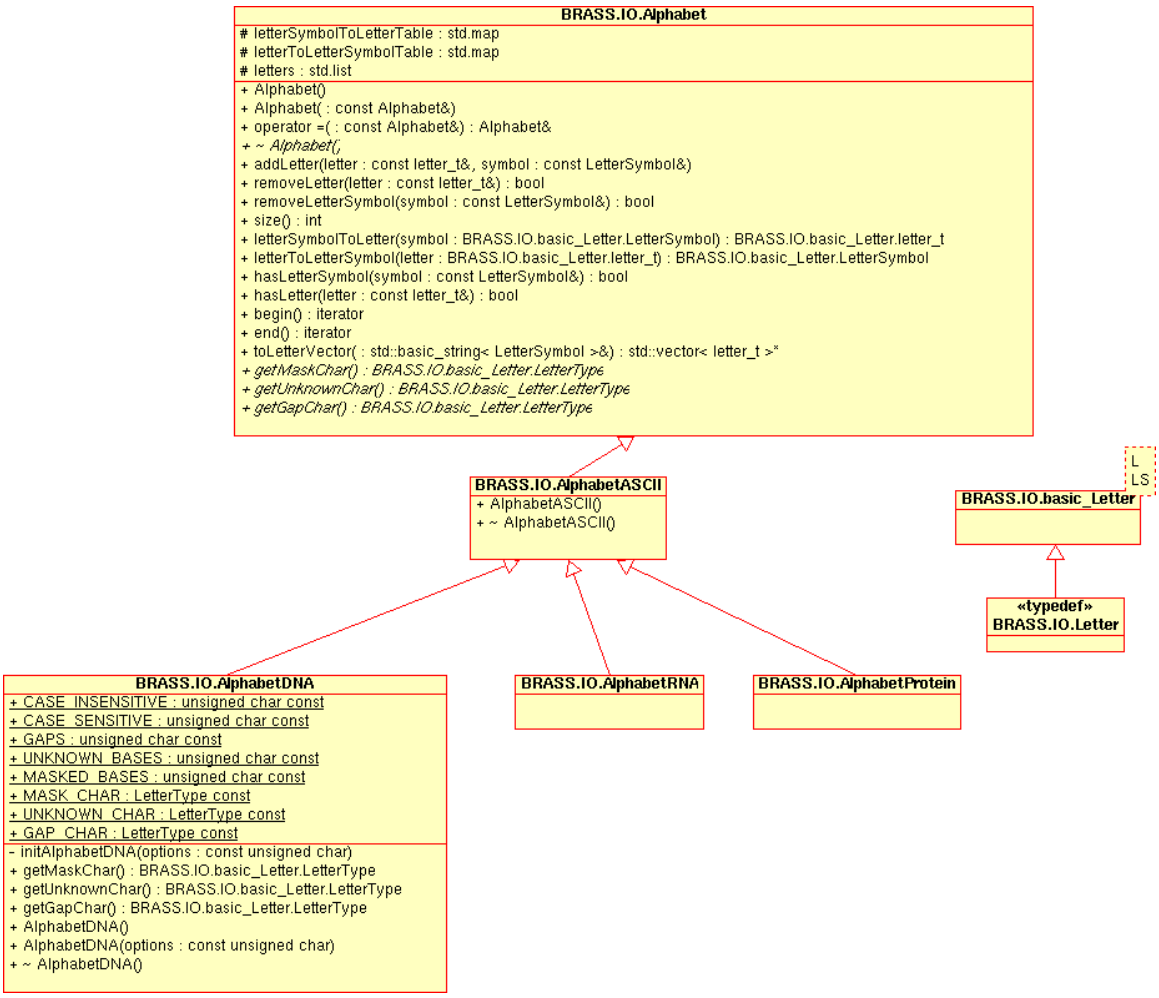


Figure A.1: BRASS::IO Core Classes

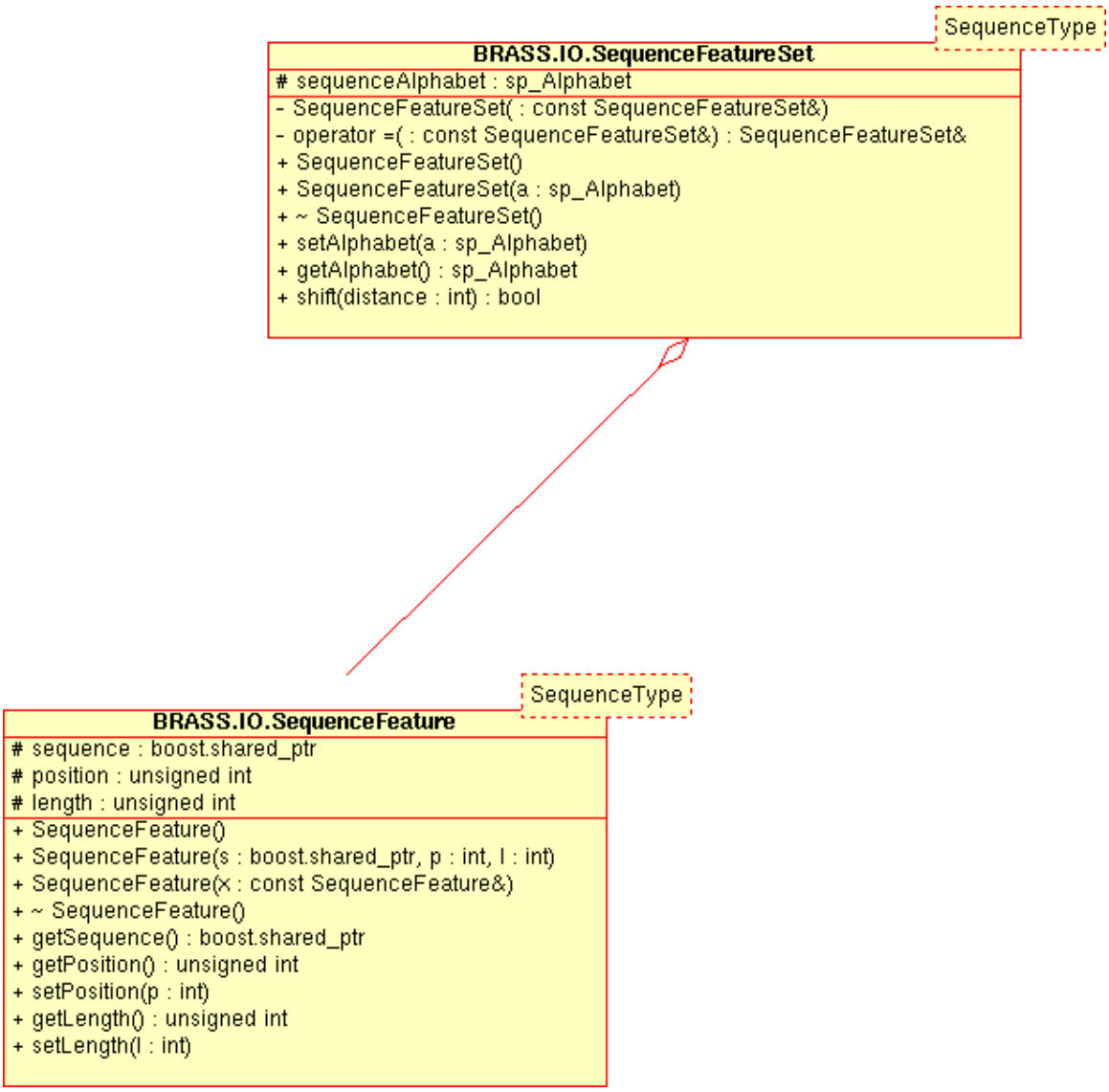


Figure A.2: BRASS::IO Annotation Classes

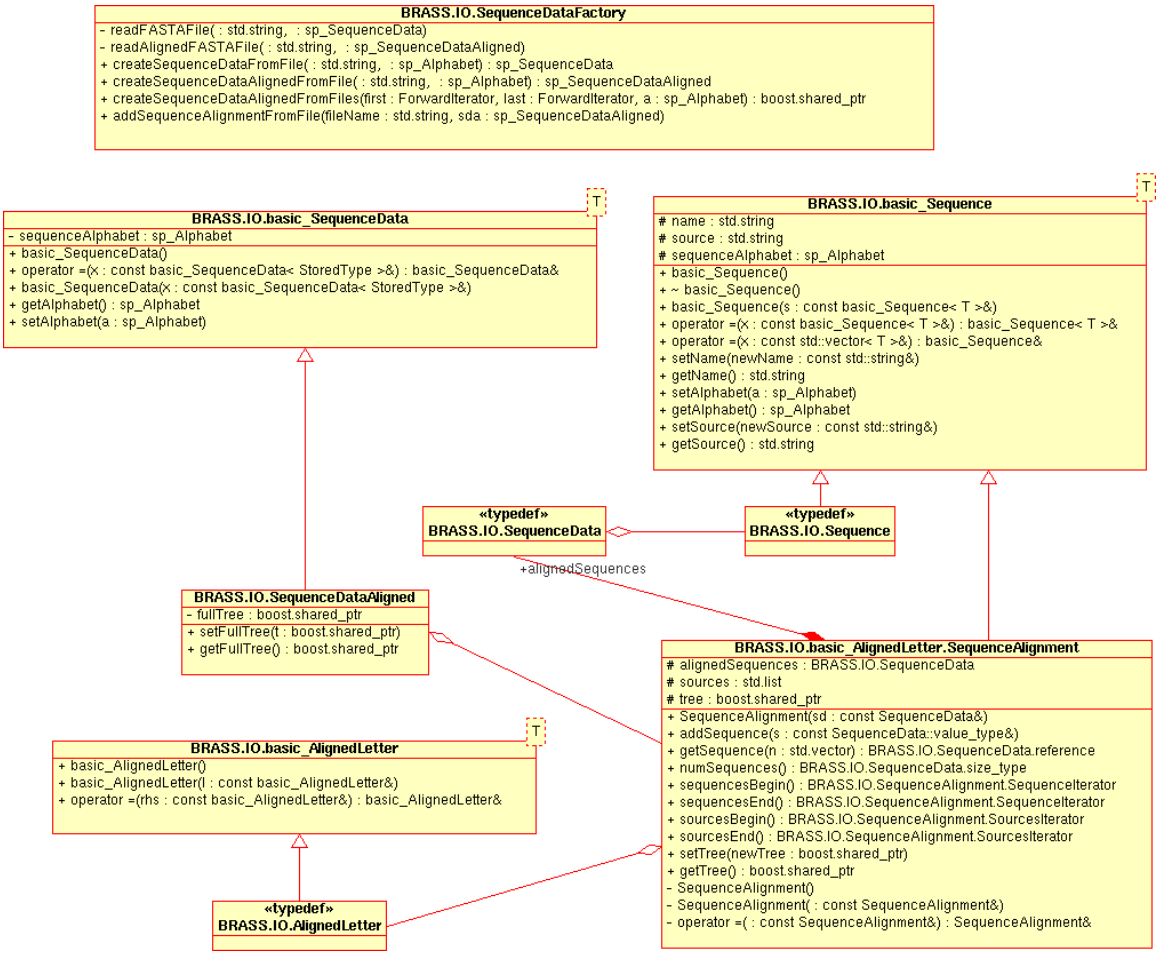


Figure A.3: BRASS::IO Sequence Classes

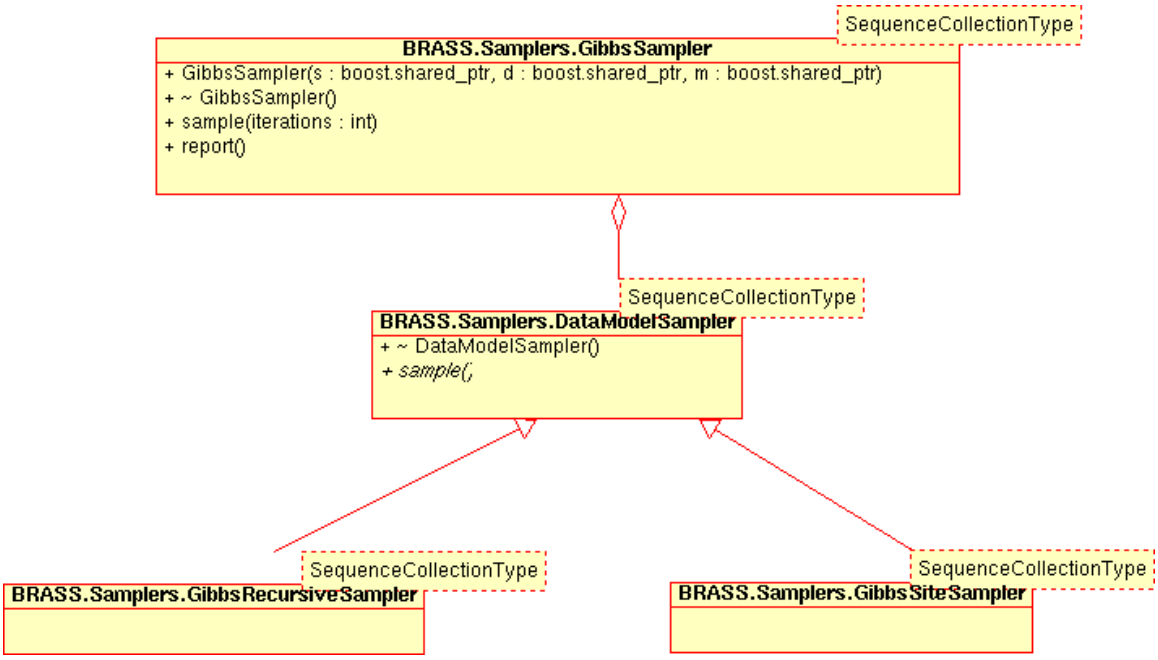


Figure A.4: BRASS::Samplers Classes

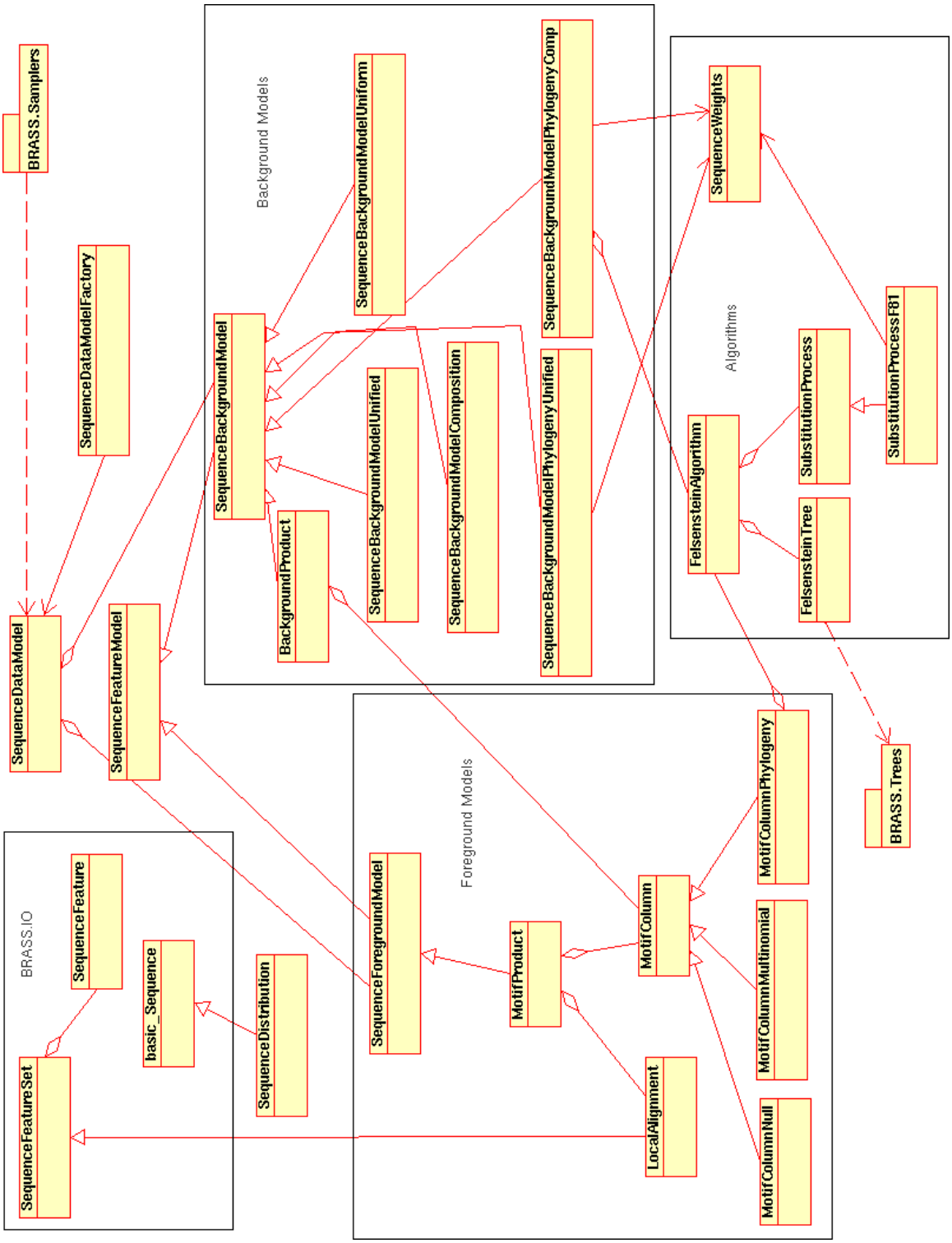


Figure A.5: BRASS::Math Overview

Figure A.6: BRASS::Math Motif Related Classes

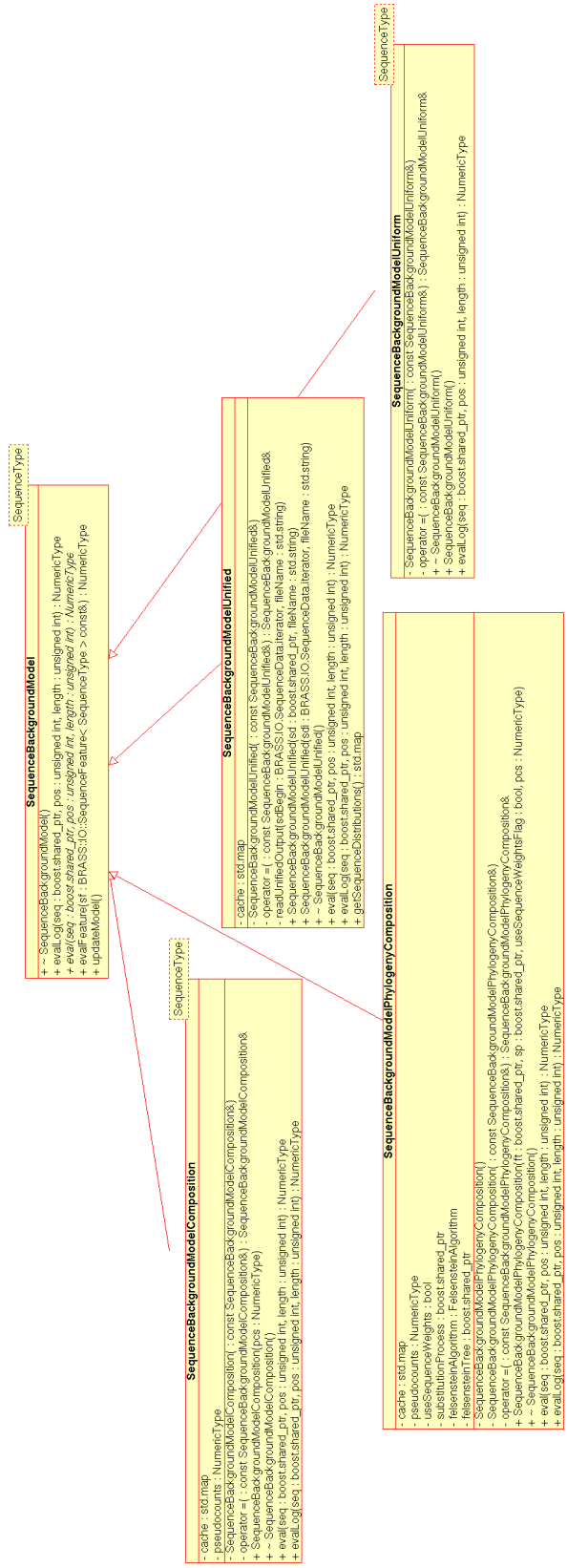


Figure A.7: BRASS::Math Background Classes

Figure A.8: BRASS::Math Factory & Models

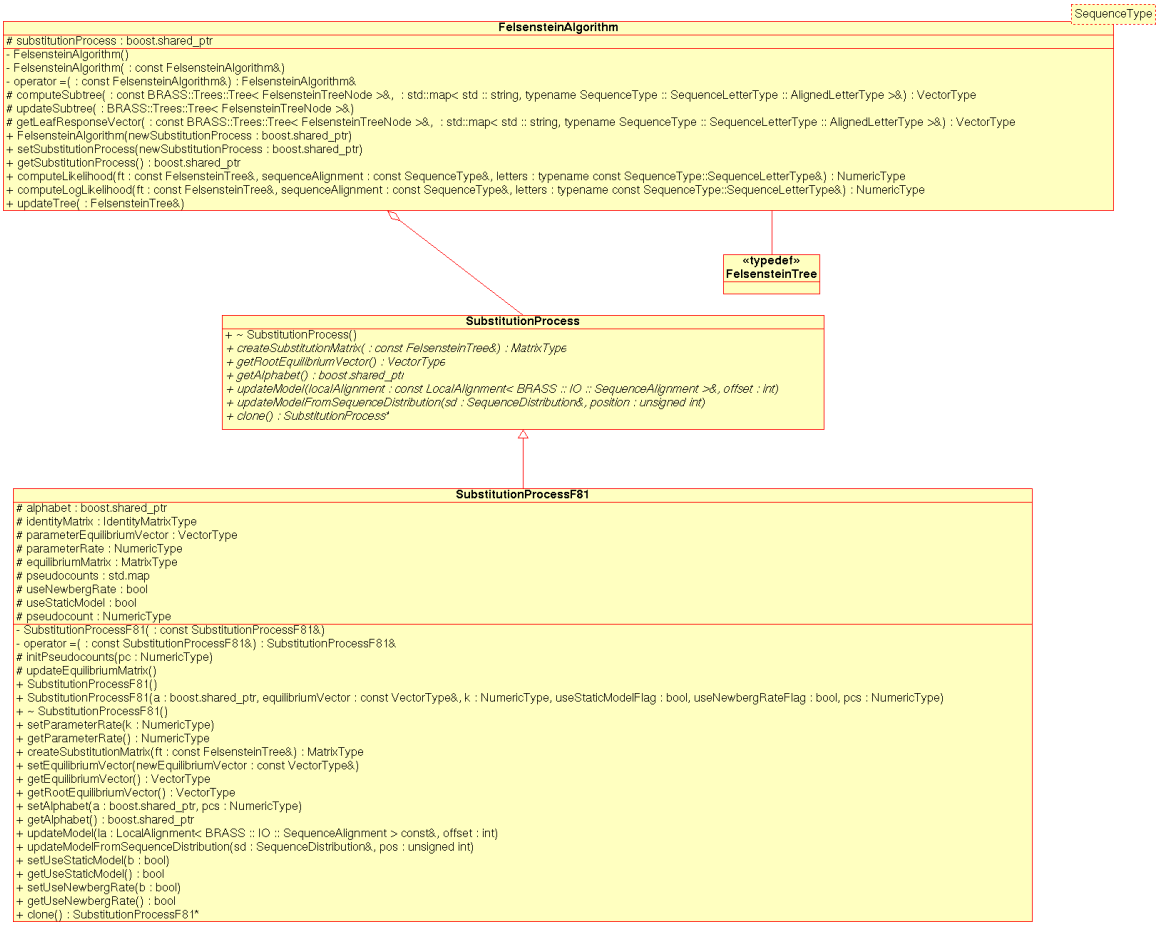


Figure A.9: BRASS::Math Felsenstein Algorithm Classes

Appendix B

BRASS Database System Supplemental Materials

B.1 ER Diagrams

The ER diagrams on the following pages illustrate the database schema.

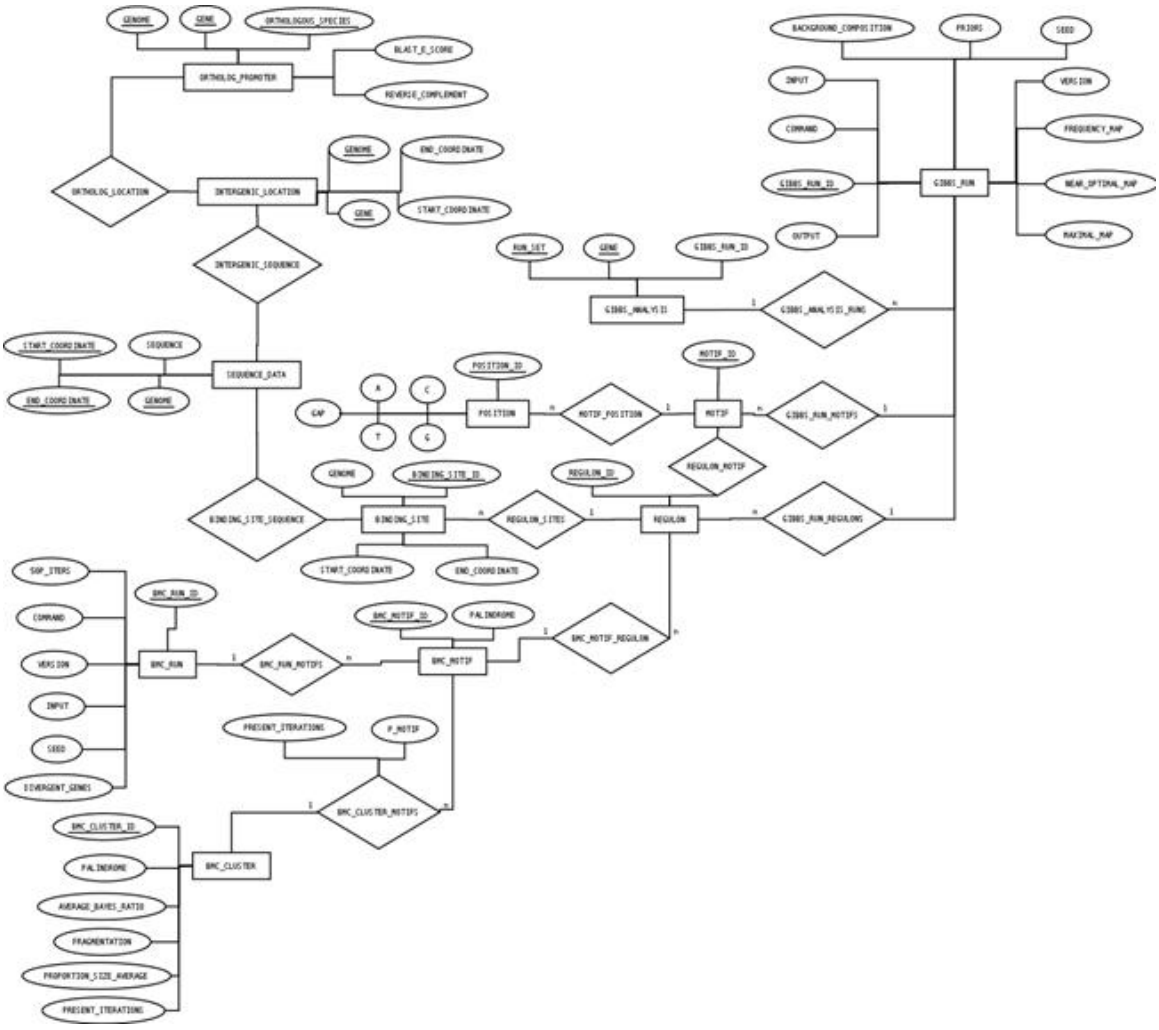


Figure B.1: Complete Database Schema (Please see detail diagrams B.2, B.3, B.4)

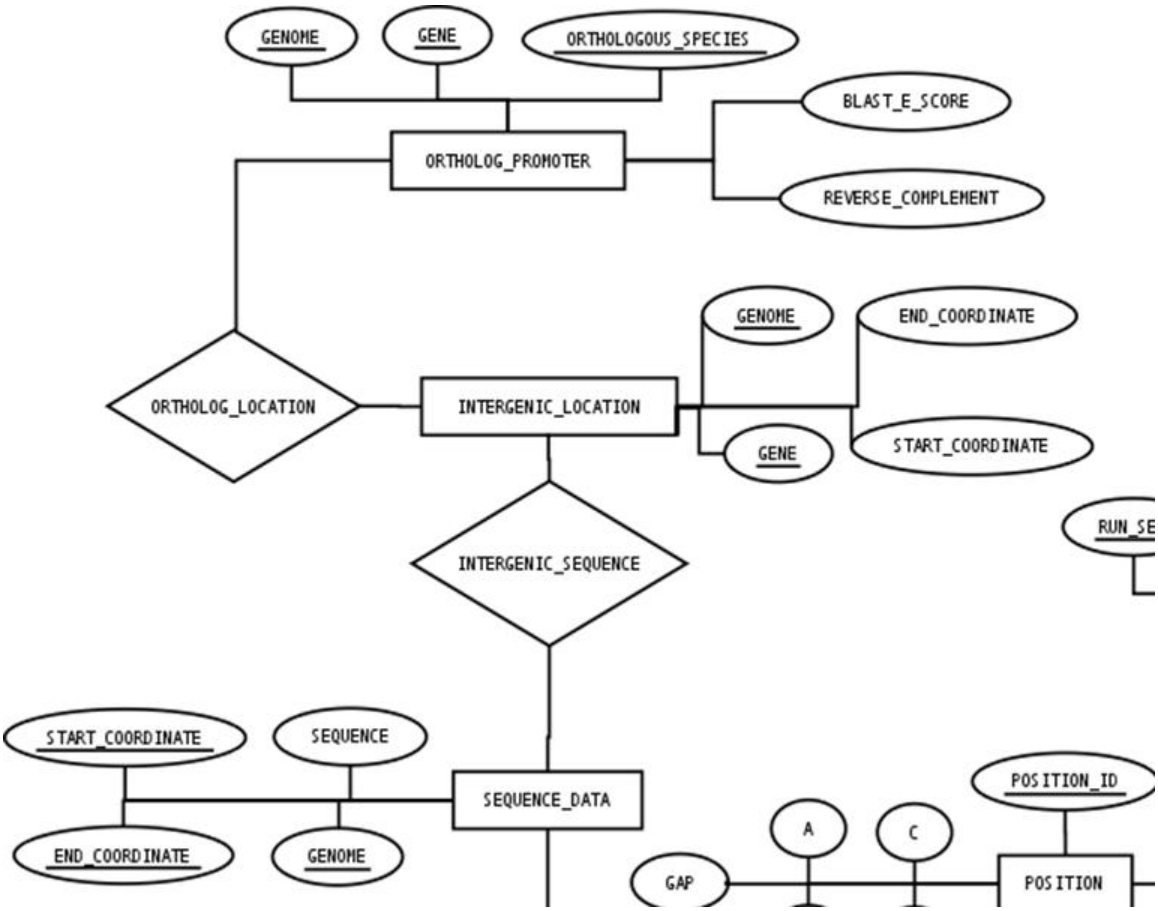


Figure B.2: Genome Shared Data Domain Schema

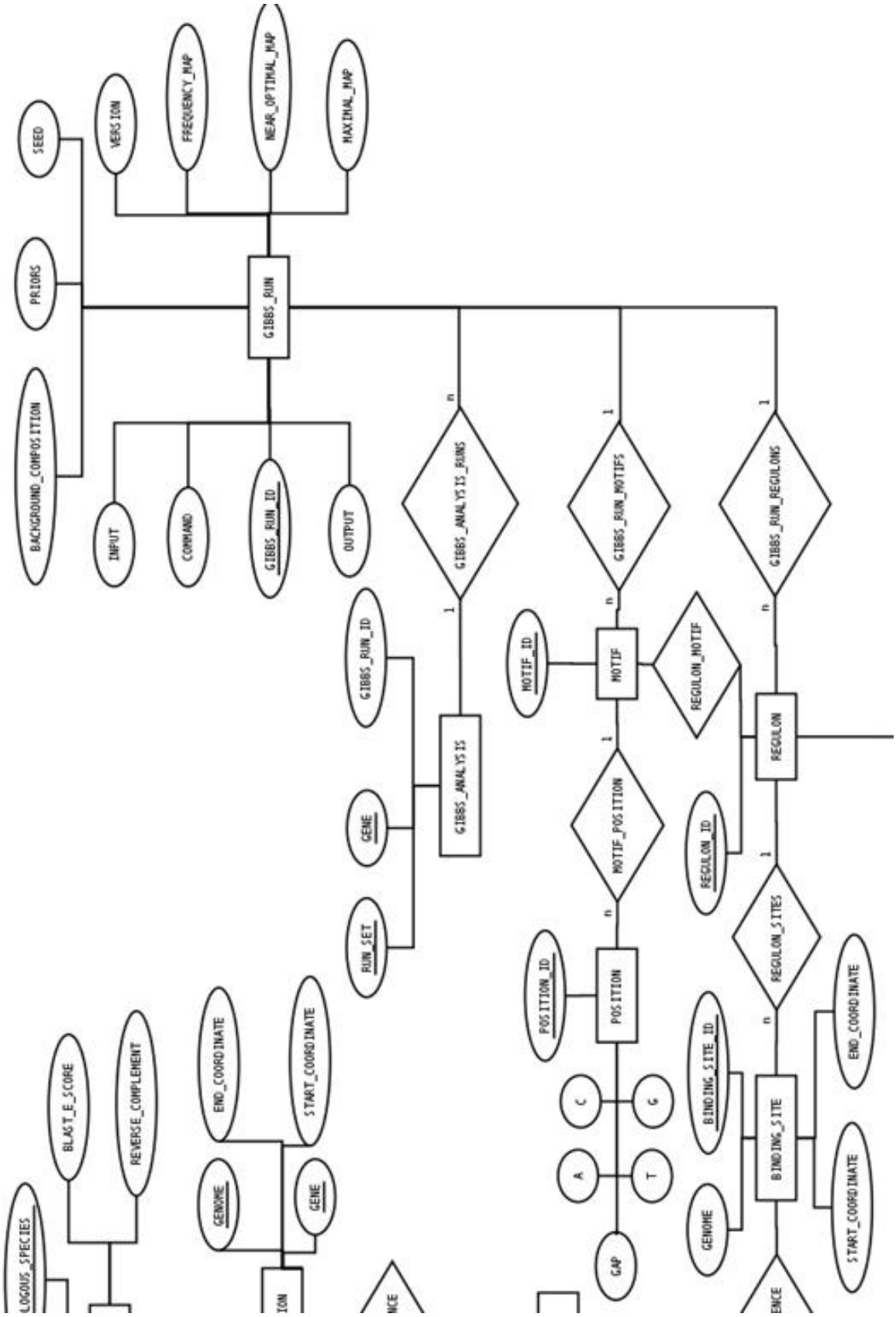


Figure B.3: Gibbs Domain Schema

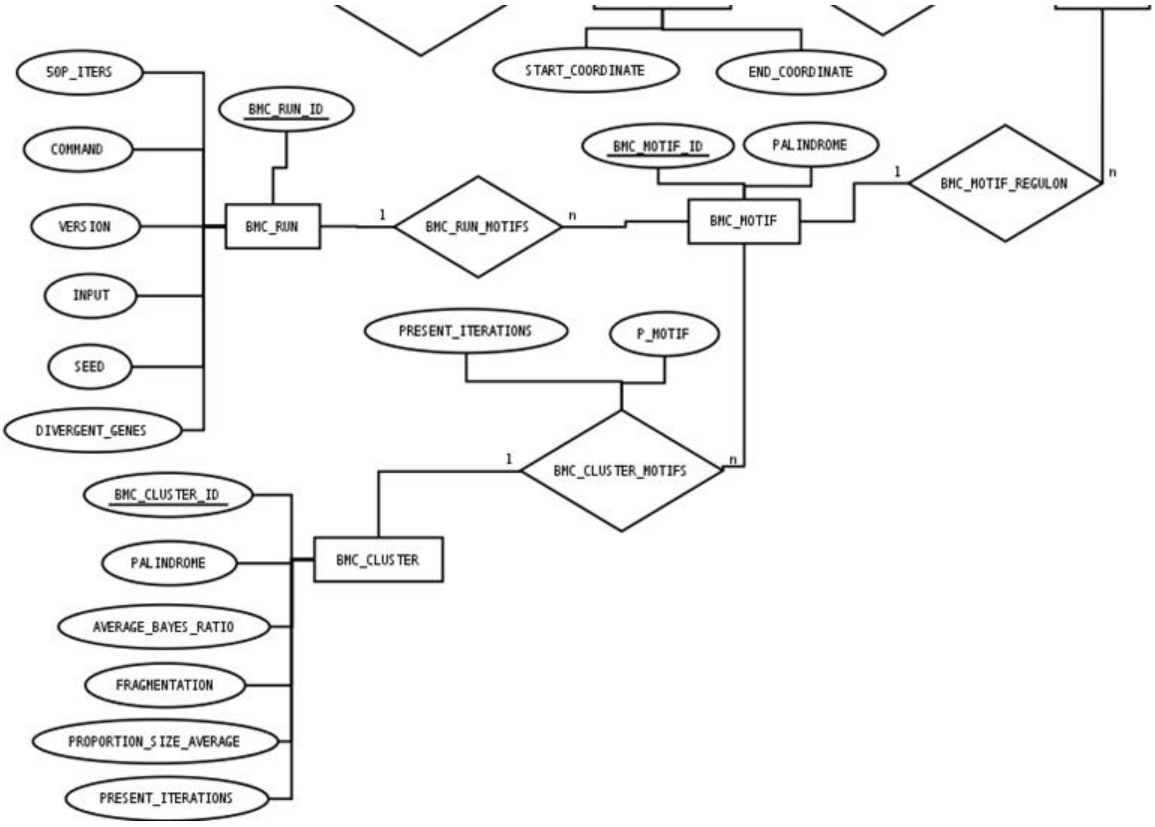


Figure B.4: BMC Domain Schema

B.2 Web Interface Illustrations

The screen captures on the following pages illustrate queries executing on the database system. Please see section 6.3.1 for more information.

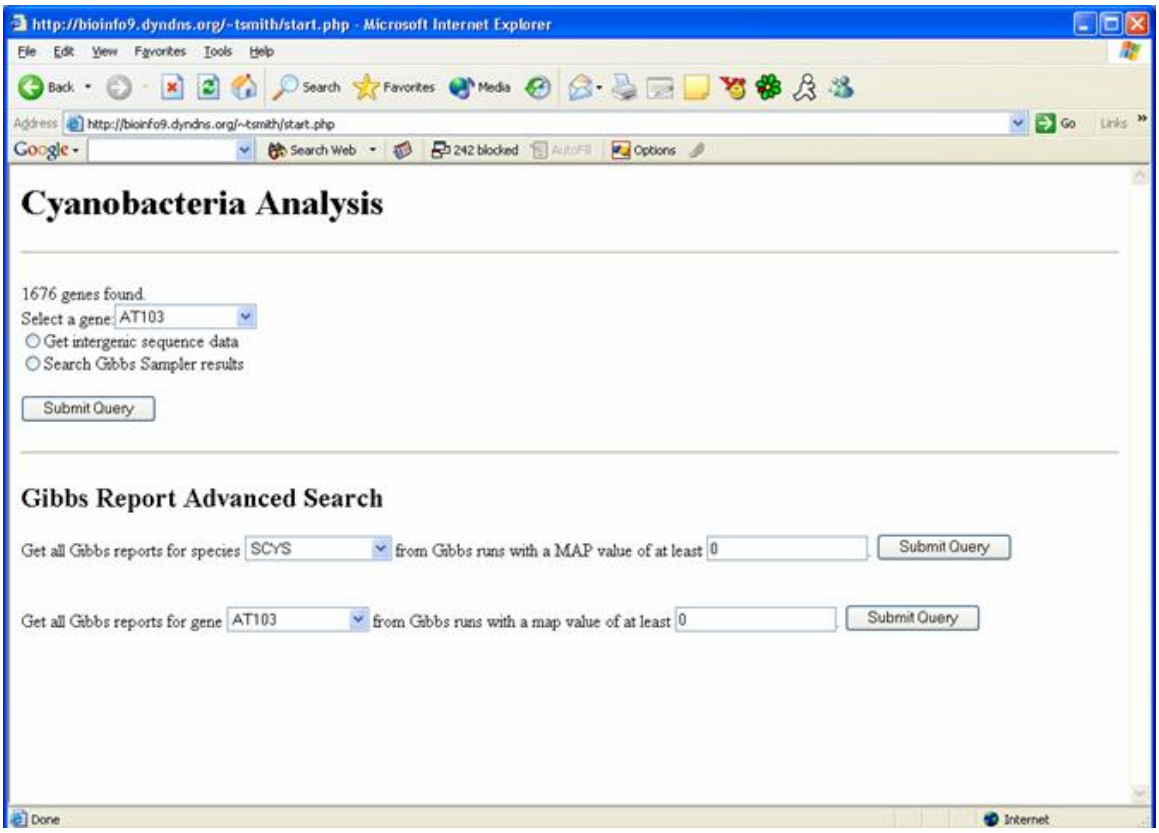


Figure B.5: The Prototype Database Query Page

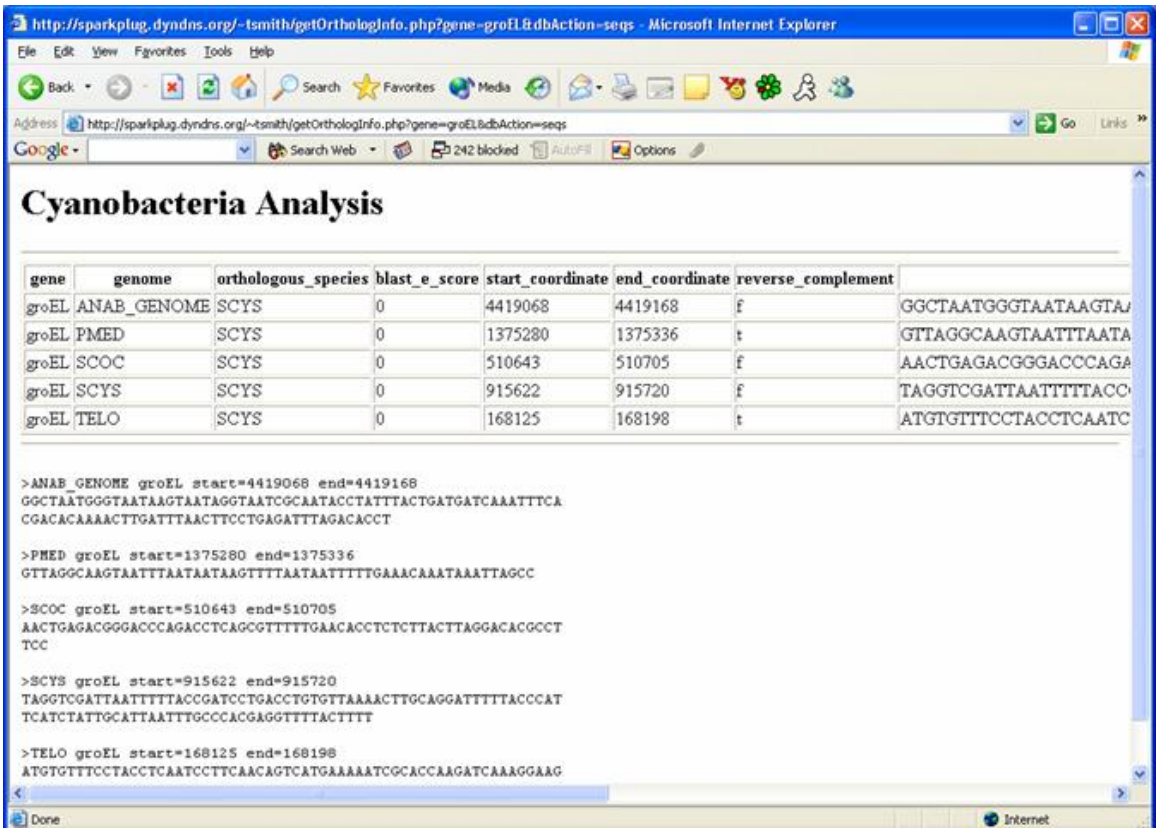


Figure B.6: Displaying Intergenic Sequence Data

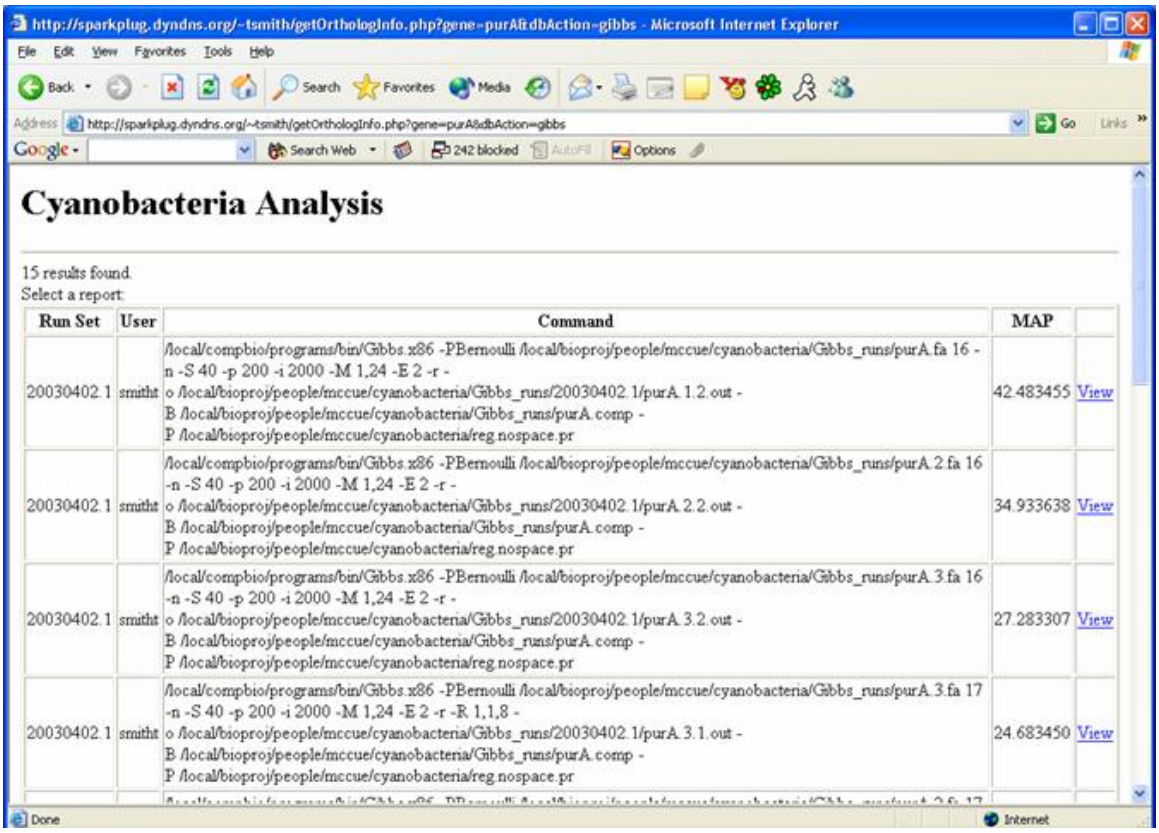


Figure B.7: Searching Gibbs Sampler Reports

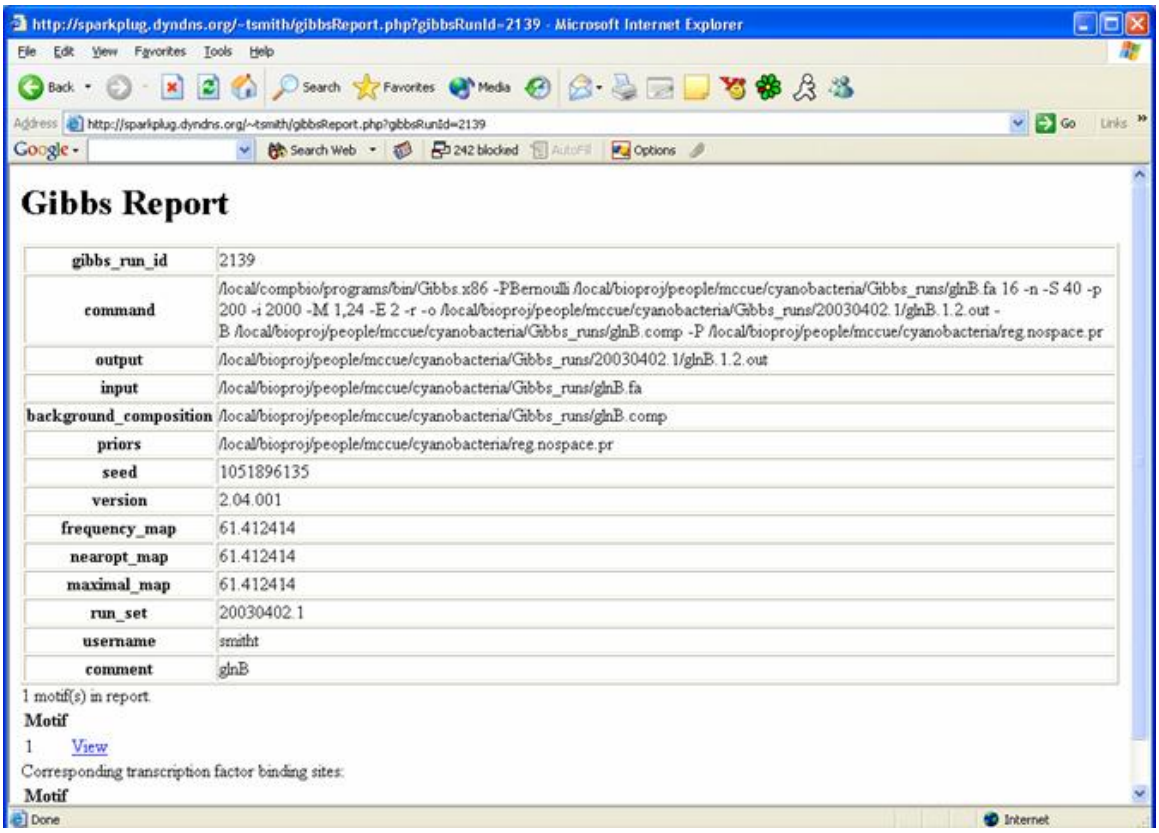
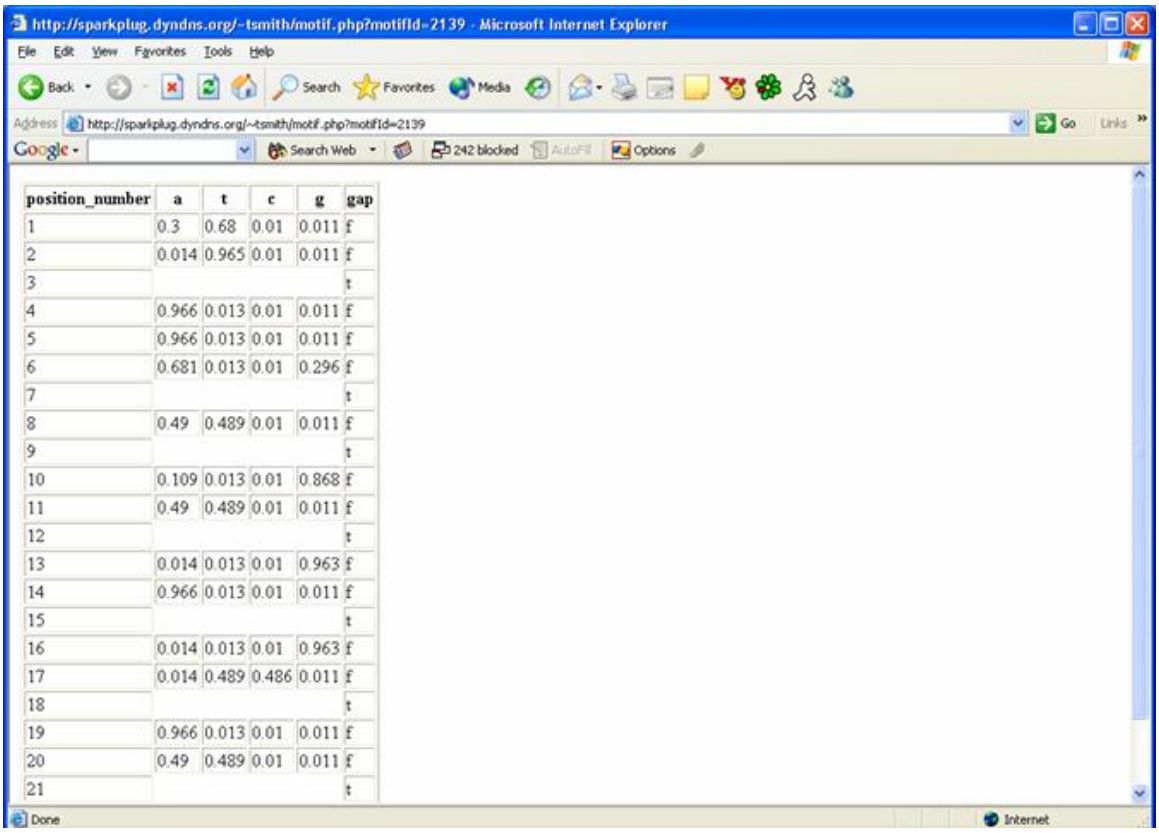
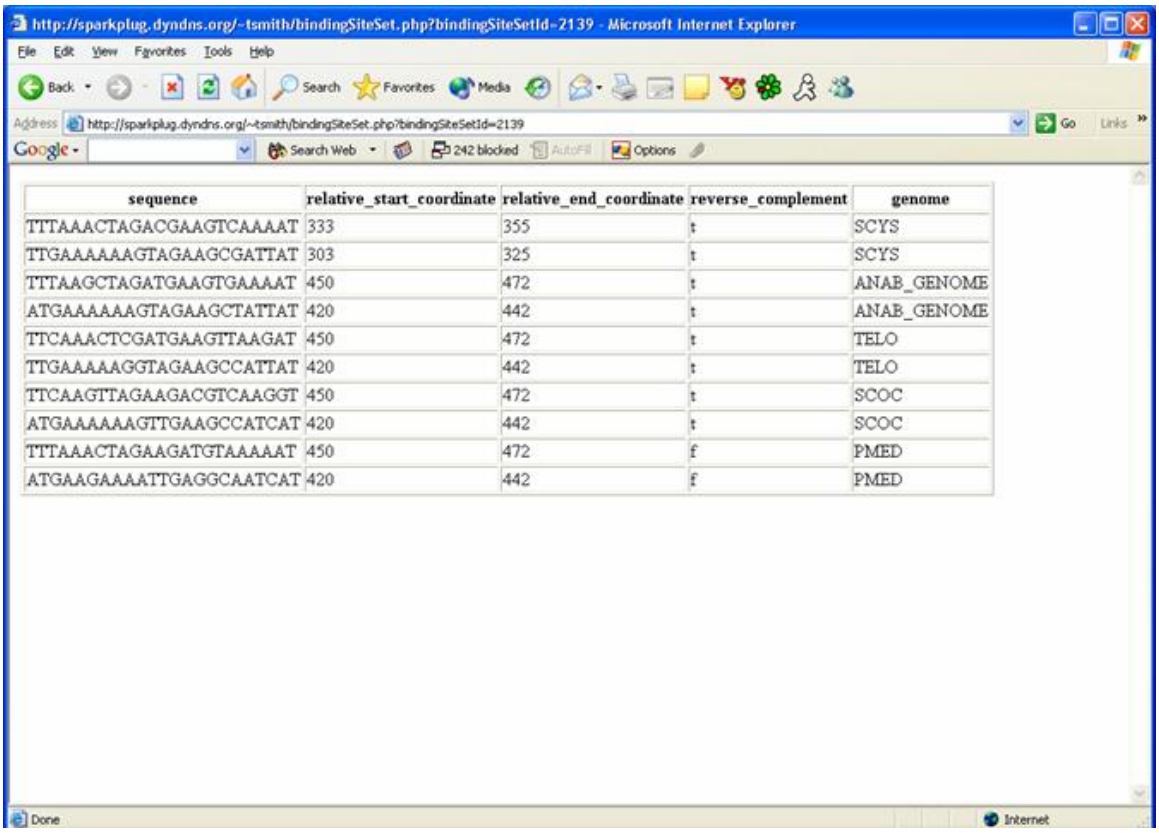


Figure B.8: Details of a Gibbs Sampler Report



position_number	a	t	c	g	gap
1	0.3	0.68	0.01	0.011	f
2	0.014	0.965	0.01	0.011	f
3					t
4	0.966	0.013	0.01	0.011	f
5	0.966	0.013	0.01	0.011	f
6	0.681	0.013	0.01	0.296	f
7					t
8	0.49	0.489	0.01	0.011	f
9					t
10	0.109	0.013	0.01	0.868	f
11	0.49	0.489	0.01	0.011	f
12					t
13	0.014	0.013	0.01	0.963	f
14	0.966	0.013	0.01	0.011	f
15					t
16	0.014	0.013	0.01	0.963	f
17	0.014	0.489	0.486	0.011	f
18					t
19	0.966	0.013	0.01	0.011	f
20	0.49	0.489	0.01	0.011	f
21					t

Figure B.9: Details of a Motif



sequence	relative_start_coordinate	relative_end_coordinate	reverse_complement	genome
TTTAAACTAGACGAAGTCAAAAT	333	355	t	SCYS
TTGAAAAAAGTAGAAGCGATTAT	303	325	t	SCYS
TTTAAGCTAGATGAAGTGAAAAT	450	472	t	ANAB_GENOME
ATGAAAAAAGTAGAAGCTATTAT	420	442	t	ANAB_GENOME
TTCAAACTCGATGAAGTTAAGAT	450	472	t	TELO
TTGAAAAAGGTAGAAGCCATTAT	420	442	t	TELO
TTCAAGTTAGAAGACGTCAAGGT	450	472	t	SCOC
ATGAAAAAAGTTGAAGCCATCAT	420	442	t	SCOC
TTTAAACTAGAAGATGTAAAAAT	450	472	f	PMED
ATGAAGAAAATTGAGGCAATCAT	420	442	f	PMED

Figure B.10: Details of Transcription Factor Binding Site Predictions