Abstract of "A database of causality-inferred structure-function information for genomic *cis*-regulatory architecture" by Ryan Tarpine, Ph.D., Brown University, May 2012.

Despite the fact that the human genome was sequenced ten years ago, there exists no database of *cis*-regulatory architecture that is validated conclusively by rigorous experimental criteria; this has been a notoriously longstanding unresolved computational biology problem. **This dissertation describes the construction of the first such database, the *cis*-Lexicon, containing only causality-inferred DNA sequence structure information on the regulatory regions of transcription factor-encoding genes and other regulatory genes**. The data in the Lexicon is purely causality-inferred, meaning that each annotation is backed by experimental techniques which prove causality; there is no information due to noisy experimental methods or computational prediction. This data, previously not available in databases (outside of the original papers) or mixed with lower-quality data, is necessary for understanding the *cis*-regulatory code, the relationship between sequence structure and regulatory function. Only through completeness of information will correct conclusions be drawn from the Lexicon, and for this purpose we built the *Cis*-Lexicon Ontology Search Engine (CLOSE). CLOSE is an information retrieval system designed to find biology journal articles containing *cis*-regulatory sequence structure information and evaluate the completeness of the *cis*-Lexicon. Information must be entered into the Lexicon in a reliable manner to ensure accurate annotations; in addition, the data must be conveniently accessible to be useful for experimental work. For both of these purposes we have built the *cis*-Browser, a genome browser customized for *cis*-regulatory analysis.

Abstract of "A database of causality-inferred structure-function information for genomic *cis*-regulatory architecture" by Ryan Tarpine, Ph.D., Brown University, May 2012.

Despite the fact that the human genome was sequenced ten years ago, there exists no database of *cis*-regulatory architecture that is validated conclusively by rigorous experimental criteria; this has been a notoriously longstanding unresolved computational biology problem. **This dissertation describes the construction of the first such database, the *cis*-Lexicon, containing only causality-inferred DNA sequence structure information on the regulatory regions of transcription factor-encoding genes and other regulatory genes**. The data in the Lexicon is purely causality-inferred, meaning that each annotation is backed by experimental techniques which prove causality; there is no information due to noisy experimental methods or computational prediction. This data, previously not available in databases (outside of the original papers) or mixed with lower-quality data, is necessary for understanding the *cis*-regulatory code, the relationship between sequence structure and regulatory function. Only through completeness of information will correct conclusions be drawn from the Lexicon, and for this purpose we built the *Cis*-Lexicon Ontology Search Engine (CLOSE). CLOSE is an information retrieval system designed to find biology journal articles containing *cis*-regulatory sequence structure information and evaluate the completeness of the *cis*-Lexicon. Information must be entered into the Lexicon in a reliable manner to ensure accurate annotations; in addition, the data must be conveniently accessible to be useful for experimental work. For both of these purposes we have built the *cis*-Browser, a genome browser customized for *cis*-regulatory analysis.

A database of causality-inferred structure-function information for genomic *cis*-regulatory architecture

by
Ryan Tarpine
B. S., Carnegie Mellon University, 2006
Sc. M, Brown University, 2009

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island
May 2012

This dissertation by Ryan Tarpine is accepted in its present form by
the Department of Computer Science as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____ _____

Sorin Istrail, Advisor

Recommended to the Graduate Council

Date _____ _____

Franco Preparata, Reader

Date _____ _____

Eli Upfal, Reader

Date _____ _____

Gary Wessel, Reader

Approved by the Graduate Council

Date _____ _____

Peter Weber
Dean of the Graduate School

Vita

Ryan Tarpine was born on September 2, 1984 in Wilmington, DE.

Education

- Ph.D. in Computer Science, Brown University, Providence, RI. May 2012.
- Sc.M. in Computer Science, Brown University, Providence, RI. May 2009.
- B.S. in Computer Science, Carnegie Mellon University, Pittsburgh, PA. May 2006.

Publications

- B. V. Halldórsson, D. Aguiar, R. Tarpine, and S. Istrail. The Clark phaseable sample size problem: long-range phasing and loss of heterozygosity in GWAS. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, 18(3):323–333, Mar. 2011.
- F. Lam, R. Tarpine, and S. Istrail. Conservative extensions of linkage disequilibrium measures from pairwise to multi-loci and algorithms for optimal tagging SNP selection. In V. Bafna and S. C. Sahinalp, editors, *Proceedings of the 15th Annual International Conference on Research in Computational Biology (RECOMB2011)*, volume 6577 of Lecture Notes in Computer Science, pages 468–482. Springer, 2011.
- F. Lam, R. Tarpine, and S. Istrail. The imperfect ancestral recombination graph reconstruction problem: upper bounds for recombination and homoplasy. Journal of Computational Biology: A Journal of Computational Molecular Cell Biology, 17(6):767–781, June 2010.
- B. V. Halldórsson, D. Aguiar, R. Tarpine, and S. Istrail. The clark phase-able sample size problem: Long-range phasing and loss of heterozygosity in GWAS. In B. Berger, editor, *Proceedings of the 14th Annual International Conference on Research in Computational Biology (RECOMB2010)*, volume 6044 of Lecture Notes in Computer Science, pages 158–173. Springer, 2010.
- J. Nam, P. Dong, R. Tarpine, S. Istrail, and E. H. Davidson. Functional *cis*-regulatory genomics for systems biology. *Proceedings of the National Academy of Sciences of the United States of America*, 107(8):3930–3935, 40 Feb. 2010.

- S. Istrail, R. Tarpine, K. Schutter, and D. Aguiar. Practical computational methods for regulatory genomics: A cisGRN-lexicon and cisGRN-browser for gene regulatory networks. In I. Ladunga, editor, *Computational Biology of Transcription Factor Binding*, volume 674 of Methods in Molecular Biology, pages 369–399. Springer Science+Business Media, LLC, Humana Press, 2010.

- R. Tarpine and S. Istrail. On the concept of *cis*-regulatory information: From sequence motifs to logic functions. In A. Condon, D. Harel, J. N. Kok, A. Salomaa, and E. Winfree, editors, *Algorithmic Bioprocesses*, Natural Computing Series, pages 731–742. Springer Berlin Heidelberg, 2009.

Acknowledgements

This dissertation would not have been possible without the help and support of many people, first of all my wife Nur and my parents, Marge and Adam Tarpine. My entire family was essential in inspiring me to put in the time and effort necessary to see this work through to completion.

Secondly, I would like to thank my advisor, Sorin Istrail. The last six years have been an invaluable learning experience, and Sorin has been an excellent mentor both professionally and personally. He is always aiming to tackle the biggest problems, whether "sexy" or not, and I hope I carry on his passion in my work moving forward.

Next, I would like to thank Eric Davidson, the foremost experimentalist of regulatory genomics and our key collaborator. He ensured that this work remained practical and focused on the needs of real biologists rather than theoretical concerns. Eric shared much of his time and wisdom with me, both over the phone and in person, and for this I am very grateful.

I certainly cannot leave out the army of annotators who went through thousands of papers to extract the information that now makes up the *cis*-Lexicon. There are too many to list here comprehensively, so I will limit myself to the team leaders: Jacob Halpert, James Hart, Kyle Schutter, and Tim Johnstone. Thank you all, for without your dedication and hours of hard work there would be no Lexicon!

Lastly, I would like to thank my colleagues whom I have had the pleasure of publishing with during my time here: Bjarni Halldórsson, Derek Aguiar, Fumei Lam, and Jongmin Nam. I am grateful for your help, your advice, and your friendship, and I have learned greatly from you.

Contents

List of Tables

List of Figures

Chapter 1

A Brief Introduction to the Regulatory Genome

Every organism starts out as a single cell that by dividing, growing, and specializing into different types of cells ultimately becomes the complex adult that we recognize—a creature with various limbs, organs, and multiple senses. The genome contains the information required to carry out these processes. It specifies the instructions on how to develop the eye, the foot, the heart, and all the other body parts and tissues.

The genome is stored in molecules called *DNA*, which are long chains of smaller molecules called *nucleotides* (also *bases* or *basepairs*). As there are four types of nucleotides, a DNA molecule in essence represents a string in a four-letter alphabet. Since the four types of nucleotides are called adenine, cytosine, guanine, and thymine, the letters in this alphabet are referred to as A, C, G, and T. One manner in which DNA stores information is by specifying how to synthesize *proteins*. Proteins are large molecules that carry out many critical processes in cells, such as metabolism, signaling, and division. Proteins themselves are long chains of smaller molecules called *amino acids*, of which there are twenty types.

The sequence of amino acids that make up each protein is stored in the genome (this is an oversimplification, but sufficient for our purposes). The manner in which a sequence is represented in DNA is called *the genetic code*. This code is very straightforward, and it has been understood for decades. Three contiguous *basepairs* of DNA (called a *codon*) are used to store each element in the sequence. For each type of amino acid, there is usually more than one codon which may encode it, but each codon always represents the same type of amino acid. For example, TTA and CTG always code for leucine and GAA always codes for glutamic acid. Three codons are known as stop codons, because they represent the end of the sequence rather than an amino acid (analogous to the EOF returned by `getchar()` in the C standard library), but every one of the other $4^3 - 3 = 61$ possible codons codes for a specific amino acid.

The genetic code explains *how* to synthesize a protein, but it does not explain *when*. This is of

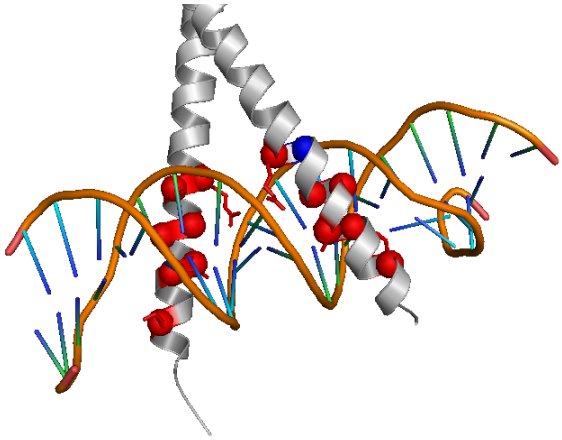


Figure 1.1: An example of how a transcription factor binds to a DNA sequence

critical importance, because every cell contains a copy of the entire genome of that organism—cells in the eye contain not only instructions on how to make the eye, but how to make the heart as well (and vice versa). However, cells in the eye "know" which orders they should be carrying out; they "know" they are part of the eye so they only develop the parts of the eye, not parts of the heart. The process of determining which proteins are synthesized and which are not is called *gene regulation*. There are many types of gene regulation; this proposal focuses on a particular one called *cis*-regulation.

Cis-regulation is caused by transcription factors binding to specific sequences in the genome (see Fig. 1.1), which trigger the activation or repression of gene expression. Transcription factors are a specific type of protein, so they themselves are encoded by genes, which must be regulated by other transcription factors. These chains of regulation yield a graph or network, known as a *gene regulatory network* (GRN). Unlike the rules governing the translation of genetic material into protein, which have been understood well for nearly 50 years, the rules governing gene expression are still only conceived of at the level of general principles. It remains difficult not only to determine the effect a given region has on gene expression, but even to recognize a *cis*-regulatory region at all.

The locations where transcription factors bind are called *transcription factor binding sites* (TFBSs). These binding sites are rarely found in sequence which encodes proteins. More often, they are found in non-coding sequence nearby the gene that they regulate, which is then referred to as the *cis-regulatory region* of that gene. TFBSs are not found scattered uniformly throughout these regions; usually they are found in clusters called *cis-regulatory modules* (CRMs). These clusters are called modules not only because they look modular—in fact their function is modular too. Each CRM performs a specific, self-contained regulatory function, which its TFBSs work together to carry out. The *cis*-regulatory region of the gene *endo16*, for example, contains six CRMs, named Modules A, B, DC, E, F and G (see Fig. 1.2). Module A ensures that the gene is expressed initially during development, Module B causes activation at a later stage, Module DC represses expression in a region where the gene should not be expressed, and the other modules perform similar but independent

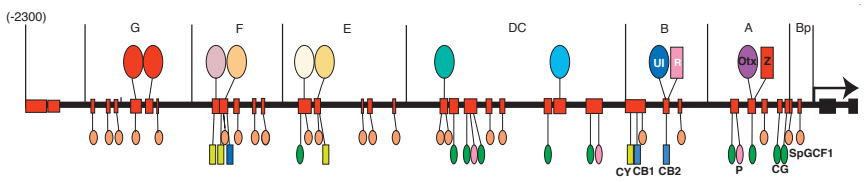


Figure 1.2: *Cis*-regulatory region of *endo16* (from [117]). The red rectangles represent the locations of TFBSs. The ovals and rectangles linked to the sequence (thick line) by thin lines represent the transcription factors. TFs above the sequence are those which bind uniquely in a single region of the sequence; TFs below bind in multiple locations.

regulatory functions. One can even see that Module DC contains two clusters of TFBSs, but these two clusters comprise one single CRM, not two. It is the function that defines the boundaries of a module, not the locations of its TFBSs.

Each individual TFBS performs a particular regulatory function, whether activation, repression, or any of the other possible functions that we will introduce later. The regulatory function of a CRM is a combination of the individual TFBS functions. The rules of how the overall function is represented in DNA comprise the *cis*-regulatory code. To crack this code we need three types of knowledge:

1. Identification of binding sites and their TFs (the **Identification Problem**)
2. Interpretation of individual site functions (the **Interpretation Problem**)
3. Rules for combining the individual functions to infer overall output (the **Combining-Rules Problem**) [24]

All three problems have proven difficult to tackle algorithmically. We discuss them below.

1.1 The Identification Problem

1.1.1 TFBS Identification

Transcription factor binding sites (TFBSs) are short and degenerate. They are typically only 6-10 bp long, and any one transcription factor binds to a variety of different sequences. For example, the transcription factor *Su(H)* (suppressor of hairless) binds at seven known sites in the regulatory region of the gene *gcm* (glial cells missing) in the sea urchin *S. purpuratus* [85]. These seven sites contain six unique sequences (see Fig. 1.3). While the sites are eight bp long, sequence variations are found at four of the eight positions (i.e., half). It is not possible to represent all of these sites with both sensitivity and specificity easily. There are two major models that attempt to capture the permissible variation accurately: consensus sequences and position weight matrices.

The concept of consensus sequences easier to understand than to define precisely. A consensus sequence is a sequence that matches all the example sites "closely, but not necessarily exactly"

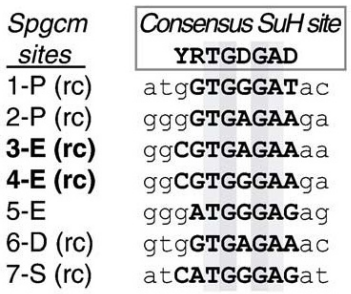


Figure 1.3: Multiple mostly distinct TFBS sequences for Su(H) in the *cis*-regulatory region of *Spgcm* (from [85])

```
TACGAT
TATAAT
TATAAT
GATACT
TATGAT
TATGTT
```

Figure 1.4: An example set of TFBSs bound by a single transcription factor [104]

[104]. It allows for variation in two ways: (1) it defines additional letters which represent specific sets of nucleotides, such as R (which stands for A or G), B (which stands for C, G or T), and N (which matches anything); and (2) it allows for a global mismatch threshold, such as allowing 1 or 2 mismatches anywhere in the pattern. [104] gives the six sequences in Fig. 1.4 as an example set of sites to observe the behavior of consensus sequences. If one tries to use TATAAT as a consensus sequence allowing no mismatches, only two of the six sites are detected, and in random sequence a nonfunctional match would be found every 4 kbp on average. If one (resp., two) mismatches is allowed, then three (resp., six) sites are detected but a occurrence will arise every 200 (resp., 30) bp on average in random, nonfunctional sequence. One can see clearly that allowing more mismatches increases sensitivity at the loss of specificity. The same conclusion is reached as well if one starts with the consensus sequence TATRNT and allows 0 or 1 mismatch. It doesn't seem possible to capture all six known functional TFBSs without ruining the specificity to the point where the model is useless. The best of the five consensus sequences appears to be TATRNT with no mismatches, which captures four of the six example sites with an average random occurrence rate of 1 / 500 bp.

The second popular TFBS sequence model is position weight matrices (PWMs). By incorporating not only the bases known to appear at each position but also their probabilities of occurrence, a much more precise model is made. While the independence/additivity assumptions are imperfect, they are a good approximation of reality, especially for the simplicity of the model. While there are a few notable exceptions, most factors are described relatively well by a PWM [11]. The logarithm of the observed base frequencies has been shown to be proportional to the binding energy contribution of the bases [12], so there is clear biological significance to using these values as the weights of

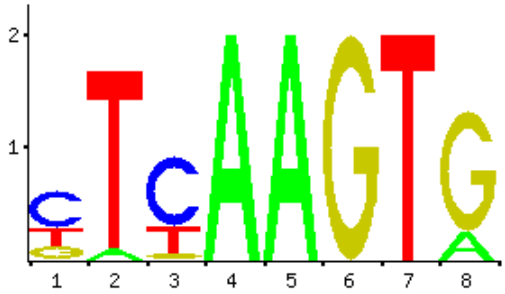


Figure 1.5: Sequence logo for *D. melanogaster tin* [84]

the matrix. But a matrix of coefficients is not sufficient to predict binding sites—there is still the question of the best cut-off score. Unlike a consensus sequence, a PWM assigns a score to every sequence, and a cut-off must be chosen as the minimum score to expect a sequence to be a functional binding site. Given a model of random sequence, it is possible to choose a cut-off score to achieve a set level of statistical significance [20] or to maximize an objective function applied to known CRMs [14]. Even with a cut-off chosen to best match the experimentally known sites, it is unclear how "good" a site is if it is barely over the limit, or how nonfunctional a sequence is if it is just under the cut-off. As true binding sites are not merely "on" or "off," but have an effect whose degree may depend on the strength of the binding [104], even the concept of a cut-off may be incorrect.

The motif recognized by a PWM is often visualized using a sequence logo [93]. Each column shows the nucleotides observed to appear in that position, in sorted order with the most common bases on top (see Fig. 1.5). The relative sizes of the letters within a column also show how often each base occurs with respect to the others. The height of each column is proportional to the information content of the base frequencies at that position. If a position always contains the same single letter, then its information content is 2 bits (the tallest possible column). If two bases are equally likely, then the position contains 1 bit of information. If all four bases are equally likely, then the information content of that position is 0 bits.

Whatever model of binding site sequences is used, the results are notoriously inaccurate. This led Wasserman and Sandelin to state their Futility Theorem:

> [E]ssentially all predicted transcription-factor (TF) binding sites that are generated with models for the binding of individual TFs will have no functional role [110]

To demonstrate this visually, we examined the regulatory sequence of the gene *endo16* from *S. purpuratus*. We searched for individual sites matching known binding site sequences, to avoid any bias that could be caused by using one of the above models. Two *cis*-regulatory modules of this gene have been studied, yielding nine unique transcription factor inputs binding to 17 sites [117, 116]. We searched for additional sequences which look like binding sites for these same nine factors within the two modules by searching for sequences identical to those we have recorded, except allowing for one

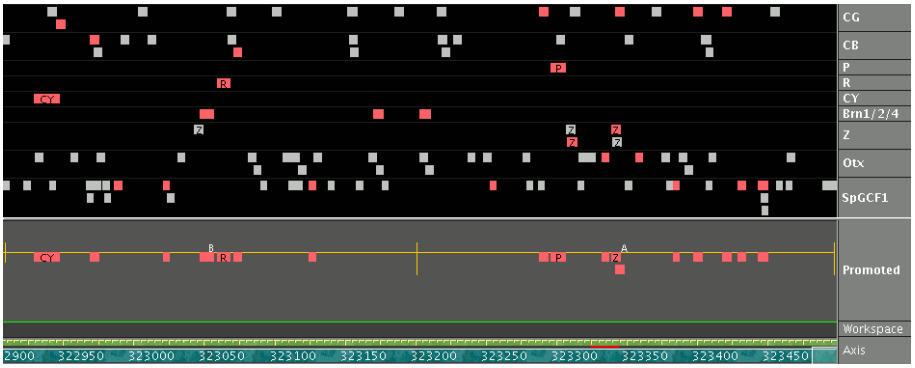


Figure 1.6: Spurious putative binding sites in the regulatory region of *endo16*

single mismatch. For one of the inputs, *otx*, we knew of four other sites in other genes (*blimp1/krox* [69] and *otx* itself [115]), and we included those in our search, yielding three unique site sequences total for *otx*. For another input, *brn1/2/4*, we knew of one other site in *blimp1/krox* [69], so we included its sequence as well. The result of our search as visualized in the *cis*-Browser can be seen in Fig. 1.6. Exact matches are highlighted in red, and matches with one mismatch are drawn in gray.

The failure of consensus sequences and PWMs implies that there is more to determining whether a site is a functional TFBS than the sequence of that site alone. The Futility Theorem is stated very carefully: it only applies to "models for the binding of individual TFs". To overcome these limitations we must analyze sites in their contexts.

1.1.2 The Motif Finding Problem

The above binding site sequence models can be useful when one has a sequence to search for sites within and one knows in advance which transcription factor to look for. However, it is often the case that the factor is unknown, so further information is needed in order to search for sites. One popular technique in this case is to look at a set of genes that appear to be coregulated, i.e., they are expressed at the same time in the same location. It is very likely that the same transcription factor regulates these genes, either directly or indirectly. If several of the genes are in fact regulated directly, many of the regulatory regions of these genes will contain a binding site for that common factor. By simply searching for a short sequence which is found to be overrepresented (i.e., more common than expected by chance), in principle we should be able to find such a binding site. This is known as the *motif finding problem*.

Unfortunately, the binding sites will probably not be identical. Some type of tolerance for mismatches must be added to the search algorithm, which complicates things considerably (otherwise a simple count of the number of occurrences of, e.g., every 8-mer would suffice). Some algorithms model the motif they are looking for combinatorially as a consensus string with a maximum number of mismatches (e.g., [83]), while others use a probabilistic or information-theoretic framework (e.g.,

[58]).

Existing motif algorithms perform reasonably well for yeast, but not for more complex organisms [22]. Several evaluations of the many proposed methods have been attempted, but the use of real genomic promoter sequences is hampered by the simple fact that "no one knows the complete 'correct' answer" [65, 106]. For an overview of the algorithms and the models they are based on, see [22].

We conjecture that the "complete correct answer" will not be known until the *cis*-regulatory code is understood. This is due to the simple fact that experimental methods can only seek to verify whether a region is functional in a certain time, location, and under specific conditions. They cannot prove that a region is never functional. Putative regulatory regions which algorithms detect may be functional but under conditions not yet discovered.

1.1.3 CRM Identification

A reasonable hypothesis that explains why looking for individual transcription factor binding site (TFBS) sequences is not sufficient is that a site is only functional if it is part of a *cis*-regulatory module (CRM). Then to identify functional TFBSs we must identify CRMs. This might sound like a chicken-and-egg problem, but fortunately this is not quite the case. There are two main approaches for identifying CRMs: (1) searching for clusters of binding sites and (2) finding conserved non-coding sequence.

What consensus sequences and PWMs capture may not be sufficient, but it is a necessary aspect of TFBSs. Therefore we can use this as a starting point for identifying CRMs. Since CRMs must contain multiple TFBSs, one straightforward model of CRMs is clusters of TFBSs, where a cluster is defined in some biologically meaningful way that is unlikely to occur in random sequence. One definition of cluster is several TFBSs for the same transcription factor within a small distance of each other (known as *homotypic* clusters), while another definition would require TFBSs for multiple transcription factors (known as *heterotypic* clusters). With both definitions some type of multiple testing correction is necessary. Few CRMs are homotypic, so the first definition captures only a small minority of the known regulatory modules. It is not feasible to search for heterotypic clusters without narrowing down the possible combinations ahead of time, because if n transcription factors are known, then there are $\binom{n}{3} = O(n^3)$ sets of three transcription factors which one could hypothetically find a cluster of. Biological insight is necessary to reduce the number of hypotheses to test.

Even when the precise transcription factors are known ahead of time, the existence of a cluster of TFBSs is not sufficient for establishing a functional CRM. In [43], knowledge of the architecture of a CRM for the gene *otx* in the sea urchin *S. purpuratus* suggested that a similar CRM should exist in the sea star *A. miniata*. The tool Cluster Buster was used to search sequence near the *A. miniata* homolog of *otx* for clusters of binding sites for *gatae*, *krox/blimp1*, and *otx* itself. The seven highest-scoring clusters with binding sites for all three factors were experimentally tested, and only the seventh cluster was found to be a functional CRM. It is not clear what distinguishes the six nonfunctional clusters from the one CRM. It is not reasonable to suggest that the six are CRMs

which function at other times, because the same transcription factors are involved—if clustering was sufficient to create a CRM, then they should all function at the same time. Section 2 discusses in detail some binding site cluster modeling and analysis that we performed, showing even more clearly the limitations of using clustering to detect *cis*-regulatory modules.

According to [63], not only is clustering insufficient, it is not even necessary. The authors point to one CRM of the *Drosophila runt* gene that is 5 kb long with TFBSs spread out diffusely over its length. While few CRMs of this type are known, this could easily be due to selection bias—most CRMs have been identified due to their TFBS clustering in the first place.

Another approach for identifying *cis*-regulatory modules is to compare the genomic sequence of related species to find conserved regions that do not code for protein. Like protein-coding sequence, regulatory sequence has a functional purpose and most mutations to it will cause harm to an organism. Therefore, few offspring who have any changes to the regulatory sequence will survive, in contrast to those who have changes to sequence outside the regulatory and coding regions, which should have no difficulty. Over generations, while a few minor changes may occur within functional regions, large changes will accumulate in the rest. By examining the sequence of species at the right evolutionary distance, we should see clear conservation only where the sequence has a specific function. We can exclude the protein-coding sequence from our analysis, either by using predicted gene models (e.g., [25]) or by transcriptome analysis (e.g., [89]), and only look the conserved patches of unknown function, which are likely to contain regulatory sequence (see, for example, [114]). For the highest accuracy, several species can be compared simultaneously [13].

This approach has one central drawback: sequence from two species of the perfect evolutionary distance is necessary. If the species are too closely related, then nearly everything will be conserved, whether functional or not. If the species are too far apart, then even the *cis*-regulatory regions will be so different as to be unrecognizable. Even when such species are known, the sequence may not be available. Genome sequencing projects have tended to focus on diverse, distantly-related species rather than thorough sampling of related species. This is changing as the cost of sequencing continues to decrease, but for the time being biologists cannot expect to have the complete genomes of closely-related species. Some researchers may have enough resources to fund their own sequencing of the particular regions that contain homologous genes they are interested in, but many do not. In these cases, the method is simply not feasible. And of course, having the sequences of species that *should* be the correct evolutionary distance apart is not a guarantee that conserved sequence will actually be found; [7] failed to find any CRMs for the gene *cyclophilin* conserved between two species that had been successfully used for other genes in the past.

When [114] carried out this technique on the *otx* gene of *S. purpuratus*, seventeen conserved regions were detected, eleven of which turned out to have regulatory function. It is not clear whether the other six are not CRMs, whether they function at other times or locations, or whether they have a function that could not be tested (such as repression, or requiring other CRMs to mediate their effects). At this time, unfortunately, it is not possible to distinguish between these cases. Therefore we cannot determine whether conservation is a necessary property of *cis*-regulatory modules. One

could even say that the concept of necessity of conservation is not well defined, because if a certain CRM was discovered to not be conserved, one could simply say that the two species were too evolutionarily distant for the analysis to be performed correctly. It is known that mutation does operate on *cis*-regulatory regions, albeit more slowly than in nonfunctional sequence [19, 8], so for nearly any CRM, its sequence will be conserved at short evolutionary distances but not for longer ones.

In summary, there is no general purpose computational method for identifying all transcription factor binding sites or *cis*-regulatory modules.

1.2 The Interpretation and Combining-Rules Problems

Once a CRM has been identified including all of its transcription factor binding sites, determining its regulatory function is still difficult. Recent work, such as training a support vector machine (SVM) on transcription factor binding profiles [120], has not improved the state of the art much beyond simply stating that a module activates expression in the same region as its key input TF(s) (for example, if *Twi* binds, then the CRM probably gives expression in the mesoderm) [87]. Since the output of a CRM is in general a complex function of its inputs, including both Boolean and continuous operations [44], a better understanding of the regulatory code is needed. [24] describes in detail the complexities of the code:

> The bottom line is that the *cis*-regulatory code specifies Boolean or discrete as well as continuous operations, all of which are directly implied in the *cis*-regulatory DNA sequence. It is probably true that *cis*-regulatory modules always execute a mix of Boolean logic operations and processing of continuous driver inputs, and their total information processing capacities can be considered the product of the unit functions mediated by the interactions at their individual target sites. Most of the individual operations the regulatory code specifies will probably turn out to be mediated by diverse transcription factors, and there will clearly be no simple one-to-one correspondence between a given functional operation and a given target site recognizing a given species of factor.

There is a fundamental difficulty that appears when attempting to predict function which cannot be overstated: the gap between sequence structure and regulatory function. Most of the outstanding successes in computational biology has been at the one-dimensional level of sequence analysis, such as BLAST [5, 6]. These are problems which can be defined and solved without recourse to the physical biology going on behind the scenes. Finding homologous sequence, when defined in terms of simple string distance, can be addressed via standard computer science algorithm design. Considering the functional behavior of molecular entities, however, brings in external dependencies: in our case, this includes time, three-dimensional space, protein-DNA and protein-protein interactions. These dependencies, which cannot be ignored, cannot be handled without data. Algorithms simply cannot be designed apart from empirical data uncovered by rigorous biological experimentation. The *cis*-Lexicon attempts to fill the void caused by the current lack of such data.

Chapter 2

Binding Site Cluster Modeling

As mentioned above, transcription factor binding site (TFBS) clustering does not seem to be a sufficient property for recognizing *cis*-regulatory modules (CRMs). It seems surprising that of the seven highest-scoring clusters found by Cluster Buster in [43], only one was a functional CRM. [43] did not describe the other six clusters, other than mentioning that they contained TFBS for all three transcription factors (TFs): Otx, Krox/Blimp1, and Gata. These three TFs were known ahead of time because they were the inputs to a CRM already found in the homologous gene of *S. purpuratus* [115]. Experimental analysis has shown that the CRMs of homologous genes often contain TFBSs for the same TFs but with different number, spacing, order, and orientation [81, 42, 18]. The only property consistently shared is the identity of the transcription factors that bind within the CRM. With this in mind, we aimed to propose a model for clusters of TFBSs to determine the likelihood of seeing clusters at least as complex as known CRMs. With such a model, we could ask the question: how likely are we to find something that appears to be a CRM—based on clustering alone—in random sequence? If such a cluster is likely and occurs often in nonfunctional sequence, it would be clear proof that clustering is not sufficient.

We began by assuming a fixed window size for TFBSs to occur within. While existing measures are known which do not depend on a fixed window size, they too have arbitrary parameters such as a gap penalty for scoring the distances between binding sites in a putative cluster [33, 34]. Assuming that sequences that appear to be TFBSs (according to a model such as consensus sequences or PWMs; see Section 1.1.1) occur uniformly at random, we can use the Poisson distribution for estimating the probability of observing a given number of TFBSs within the window. While the Poisson distribution does introduce some error because it assumes that the TFBSs are independent and that overlapping does not matter [14], this does not affect the type of CRMs we are interested in at this point, because the ratio of binding site sequence to CRM sequence is quite low (e.g. 9 TFBSs in 500 bp [115]). The Poisson distribution can compute the probability for a single window, but it does not extend to handling a long sequence scanned by a sliding window. This behavior is captured by a *scan statistic*.

2.1 Scan Statistics

For the simplest case, when there is a single transcription factor to consider, we represent an L bp DNA sequence as a sequence of L Bernoulli random variables $X_1, X_2, \ldots, X_L$ where $X_i = 1$ iff there is a TFBS at position i. For a TF whose sites appear on average every R bp, the probability of success for each X is $p = 1/R$. For example, for TFBSs which occur every 1 kbp on average, $R = 1000$ so $p = 0.001$.

We are interested in determining, for a given window size W, the threshold k such that the probability of seeing k sites in any window of the sequence is less than some significance level, such as 0.01. We will extend this to multiple transcription factors later. The maximum number of binding sites found in any window of a long sequence is technically known as the *scan statistic* S_W. It is defined in the following manner: Let Y_i be the number of TFBSs in the window starting at position i. Then

$$\begin{aligned} Y_i &= \sum_{j=1}^{W} X_{i+j-1} \\ S_W &= \max_{1 \le i \le L-W+1} Y_i \end{aligned}$$

We want to find k such that $P(S_W \ge k) < 0.01$. Since the TFBSs are uniformly distributed, the Y_is are identically distributed Poisson(pW) random variables, and it is easy to calculate $P(Y_i \ge k)$. Unfortunately, it is not easy to calculate $P(S_W \ge k)$. The strategy that first comes to mind is to try:

$$\begin{aligned} P(S_W \ge k) &= 1 - P(S_W < k) \\ &= 1 - P(Y_1 < k \wedge Y_2 < k \wedge \cdots \wedge Y_{L-W+1} < k) \end{aligned}$$

But this derivation cannot continue. It cannot be transformed into $1 - P(Y_1 < k)P(Y_2 < k) \cdots P(Y_{L-W+1} < k)$ because the Y_is are not independent. To see this, note that if $Y_i = a$, then Y_{i+1} can only be $a - 1$, $a + 1$, or a, depending on whether sliding the window loses or gains a site (or continues to contain the same sites).

In fact, there is no simple formula for S_W. Scan statistics remains an active area of research (see [37] for on overview). All known exact formulas are computationally intensive and do not scale to situations we are interested in, such as window sizes in the hundreds. Several approximations have been discovered, many of which we have implemented and compared to our own method as well as simulated results.

2.2 Scan statistic approximation methods

2.2.1 Naus' Heuristic

[78] gives an exact formula for computing the distribution of the scan statistic, but it is infeasible for

large windows and sequences (examples in [37], for example, only show window sizes up to 20 and a total sequence length of up to 500). The same author later published a heuristic in [79] which he noted was "remarkably accurate." The motivation is as follows: rather than computing $P(S_W \geq k)$ directly, instead analyze the equivalent $P(S_W < k)$. Divide the sequence (of length L) into L/W non-overlapping sections of length W. Let E_i be the event that no window which starts in the ith section contains k sites. That is, $E_i = \max_{W(i-1)\leq j\leq Wi} Y_j < k$. Then $P(S_W < k) = P(E_1 E_2 \cdots E_{L/W-1})$.

$$\begin{aligned}
P(S_W < k) &= P(E_1 E_2 \cdots E_{L/W-1}) \\
&= P(E_1)P(E_2 \mid E_1)P(E_3 \mid E_1 E_2) \cdots P(E_{L/W-1} \mid E_1 E_2 \cdots) \\
&\approx P(E_1)P(E_2 \mid E_1)P(E_3 \mid E_2) \cdots P(E_{L/W-1} \mid E_{L/W-2}) \\
&= P(E_1) \prod_{i=2}^{L/W-1} P(E_i \mid E_{i-1}) \\
&\approx P(E_1) \prod_{i=2}^{L/W-1} P(E_2 \mid E_1) \\
&= P(E_1) \left[P(E_2 \mid E_1)\right]^{L/W-2} \\
&= P(E_1) \left[P(E_1 E_2)/P(E_1)\right]^{L/W-2}
\end{aligned}$$

This derivation uses two observations. First, the E_is have a useful Markov-like property: $P(E_i \mid E_{i-1}E_{i-2}\cdots E_1) \approx P(E_i \mid E_{i-1})$. Second, $P(E_i \mid E_{i-1}) \approx P(E_2 \mid E_1)$ for all $i \geq 2$.

Let $a = P(E_1)$ and $b = P(E_1 E_2)$. Since they are independent of L, they can be calculated knowing only k, W and p. Formulas are given in [79] based on theorems proven in [78]. Then

$$P(S_W < k) \approx a(b/a)^{L/W-2}$$

When we refer to Naus' heuristic below, we mean this formula.

2.2.2 Poisson Clumping Heuristic

The Poisson clumping heuristic is proposed in [3]. It recognizes the dependence between neighboring Y_is and attempts to "declump" them in order to reach a standard Poisson distribution. The general idea starts from the recognition that if a cluster exists at position i (i.e. the window starting at i contains k sites, so $Y_k \geq k$), then it is likely that the neighboring positions also contain clusters ($Y_{k-i} \geq k$ and $Y_{k+i} \geq k$ for small positive i). If we can count each "clump" of Y_is which are $\geq k$ as a single event, then these events are modeled well as a Poisson process with some parameter λ (so that $P(S_W \geq k) = P(\text{Poisson}(\lambda L) > 0)$.

The key is to determine the expected size of a clump—since we can easily calculate the expected number of Y_is $\geq k$, dividing this by the clump size leads us to the actual number of clumps. [3] estimates the clump size by modeling the window as it slides across the sequence as a random walk. The idea is that if the current window contains m sites, then the probability of losing a site by sliding

the window to the right is m/W. The probability of gaining a site by sliding the window to the right is always p. Therefore, the probability of the number of sites in the window increasing by sliding it to the right is $p(1 - m/W)$ (gaining a site while not losing a site) while the probability of the number decreasing is $(1 - p)(m/W)$ (not gaining a site and instead losing one). The probability of gaining a site is less than the probability of losing a site, so the random walk is transient. Using the transition probabilities one can calculate the expected amount of time until the walk never returns to its origin—this is the clump size.

2.2.3 Finite Markov Chain Imbedding

The finite Markov chain imbedding (FMCI) technique (see [35] for a detailed survey) models the scan statistic as a Markov chain whose state represents two features: both the exact sequence of values within the current window, and the maximum number of sites seen within any window so far. $P(S_W \geq k)$ is then the probability of entering a state representing k sites have been seen.

Consider a simple example of finding the probability of seeing four 1s in a window of size five in a sequence of length 20 ($k = 4$, $W = 5$, $L = 20$). Let the probability of seeing a 1 be p. The state 01001(2) would represent the fact that the current window contains 0, 1, 0, 0, and 1; and the maximum number of sites seen so far is two. There are two possible transitions from this state: one is to 10011(3) with probability p (gaining a site, hence the 1 on the right end of the window) and the other is to 10010(2) with probability $1 - p$ (losing a site, hence the 0).

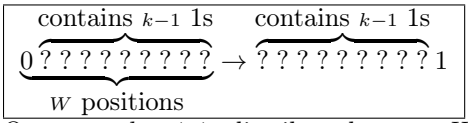
The problem definition (in terms of k, W, and p) sets the transition probabilities, defining the Markov transition matrix. All states which imply that at least three sites have been seen in a window so far, such as 00111(3) and even 00000(3), are "goal" states. The probability of being in any of these states after L transitions is precisely $P(S_W \geq k)$. This can be calculated by taking the transition matrix to the Lth power, giving an exact result. The problem is that the number of states in the Markov chain explodes exponentially as window sizes are increased; more than $\binom{W}{k}$ states are required, and for $W = 300$ and $k = 5$ this is already 1.96×10^{10}.

FMCI actually computes the exact answer, and thus is not a heuristic, but it is more practical than the algorithm of [78] because its complexity scales linearly with L. Unfortunately, it does not scale well in terms of W and k. We did not implement or test this method because it is infeasible for our problem size. All mentions of the "Markov chain heuristic" later in this document refer to the next heuristic, which we developed.

2.2.4 Our Markov Chain Heuristic

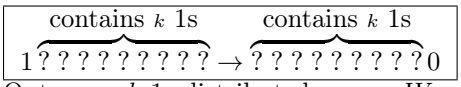
We invented our own Markov chain-based heuristic by combining ideas from the Poisson Clumping (PC) and finite Markov chain imbedding (FMCI) techniques. We first created a Markov chain based on the same assumptions as PC's random walk, letting each state represent just the number of sites in the current window. We take the transition matrix to a high power as in FMCI to estimate the probability of seeing a cluster in a long sequence. The probability of observing a cluster of k sites is

(a) State $k-1$ to state k



Outcome: $k-1$ 1s distributed among $W-1$ positions

(b) State $k+1$ to state k



Outcome: k 1s distributed among $W-1$ positions

Figure 2.1: Different causes for entering the same state in the first Markov chain heuristic model

equal to the probability of ever reaching state k. This formulation immediately extends to more than one transcription factor, just by taking a lattice of states rather than a linear sequence (where the dimension of the lattice is equal to the number of TFs). For example, the state (2,3) would represent having exactly 2 A sites and 3 B sites in the current window. The probability of moving to state (2,4) would be $p_B(1-5/W)$ (the probability of gaining a B site and not losing any of the 2+3=5 sites in the current window). The probability of moving to state (1,3) would be $(1-p_A-p_B)(2/W)$ (the probability of not gaining an A site or a B site and losing one of the 2 A sites in the current window).

We found that this heuristic was accurate for clusters of many binding sites per transcription factor but not for clusters with only a few binding sites (see the evaluations section below for details). This is important because most clusters will have several TFs but few TFBSs for each. We improved upon this model by recognizing that each state actually represents two very different cases: (1) when state k is reached from state $k-1$, the current window must contain a 1 in the rightmost position and the $k-1$ other 1s may exist in any of the other $W-1$ positions; and (2) when state k is reached from state $k+1$, the current window must contain a 0 in the rightmost position and the k 1s may exist in any of the other $W-1$ positions (see Fig. 2.1). The Markov chain transitions to state $k-1$ from k when the window contains a 1 in the leftmost position which is lost as the window slides to the right. The probability of a 1 being in the leftmost position depends on the density of 1s in the entire window (naively, the probability of transitioning from k to $k-1$ is $(1-p_{any})(k/W)$, where p_{any} is probability of occurrence of any transcription factor). Case 2 has a higher density of 1s at the left end of the window than case 1, so it seems prudent to model these two cases by different states.

We represent the state reached by having k 1s in the window after previously having $k-1$ as $\nearrow k$, and $\searrow k$ as the alternative (i.e., coming from $k+1$). A simple guess as to the transition probabilities in this new model suggests

$$P(\searrow k-1 \,|\nearrow k) \quad = \quad (1-p_{any})\frac{k-1}{W-1}$$

$$P(\searrow k-1 \mid \searrow k) \quad = \quad (1-p_{any})\frac{k}{W-1}$$

As the ratio between these two formulas is $\frac{k-1}{k}$, one can see that the difference decreases as k increases, which is why the naive model that doesn't distinguish between these two cases still performs well for high k. However, even these formulas do not match up with what is observed in practice through simulation. This is due to interesting behavior as to the loss of TFBSs as a window slides across the sequence. The time until a TFBS is lost is modeled by a type of extreme value distribution with a strict upper bound. When a binding site enters a window at its rightmost end, we are guaranteed that this TFBS will exit out the left side after exactly W transitions. If there are multiple binding sites within the window, the leftmost one will exit first, whose position is governed by an extreme value distribution. Whether or not the Markov chain transitions to a lower or higher state depends on whether a new TFBS enters the right side first, which is a random event without an upper bound. The interaction between the bounded- and unboundedness of the two possibilities with very different distributions leads to nontrivial transition probabilities. We explain our method of calculating these probabilities in the next section. We refer to this as the "tuned Markov chain heuristic" in the evaluations section below.

2.2.5 Transition Probability Calculations

Here we outline our method for calculating the "tuned" Markov chain heuristic transition probabilities for the case of one transcription factor. It is straightforward to extend this to multiple TFs.

Consider the general case of a window of size W after entering the state $\searrow k$. The rightmost end of the window is known to contain a 0 and the other $W-1$ positions contain k ones. Sliding the window to the right in the DNA sequence will eventually cause a known TFBS to fall out and/or a new TFBS to enter the window. There are three possibilities:

1. One of the k 1s might fall out of the left end of the window, making the Markov chain transition to state $\searrow k-1$
2. A new 1 might enter the right side, causing a transition to $\nearrow k+1$
3. Both of these events might happen simultaneously, causing a transition to $\nearrow k$ (since there is then a 1 in the rightmost position)

The first possibility can occur after any number between one and $W-k$ steps, depending on the location of the leftmost 1 out of the k 1s present in the first $W-1$ positions of the window. This location is modeled by the minimum value of a sample of size k where each value is taken independently from the discrete uniform distribution with range $[1, W-1]$ (this is not a perfect model since the locations of the 1s cannot coincide and thus are not independent, but it seems to be quite accurate).

The second possibility is modeled by a geometric distribution with parameter p.

The third possibility is a straightforward product of the first two happening simultaneously.

The way that each possibility can take place generates a specific sequence of Markov chain state transitions. For example, Possibility-1 occurring after 5 transitions means that the self transition $\searrow k \rightarrow \searrow k$ was taken 4 times followed by the transition $\searrow k \rightarrow \searrow k-1$. The probability of this is approximately $\left((1-\frac{4}{W-1})^k - (1-\frac{5}{W-1})^k\right)(1-p)^5$ (the probability of the leftmost 1 in the window was in the fifth position times the probability that no new TFBS entered the window in those five steps). Since we can calculate the probability of each such sequence, we can calculate the expected number of Markov chain state transitions. We then normalize these expected counts to get our transition probabilities.

Take for example the case where the window size $W = 300$, the probability of a new TFBS occurring is $p = 1/1000$ and $k = 2$. The naive transition probability for $\searrow k \rightarrow \searrow k-1$ is $\frac{k}{W-1}(1-p) = \frac{2}{299} \cdot 0.999 \approx 0.00668$. By simulation, the correct probability is estimated to be 0.00979. This seems unusual, considering it's very close to $\frac{k+1}{W-1}(1-p) \approx 0.00993$, but additional iterations of the simulation show that it is definitely different (and also not equal to $\frac{k+1}{W}$ or other similar formulas) According to our more precise method, the transition probability is approximately 0.00974. The difference is very significant when when we calculate the probabilities of long paths.

2.2.6 Approximations Evaluations

We evaluated each heuristic in comparison with estimates based on simulation via 10^5 randomly generated sequences of Bernoulli random variables. The results can be seen in Fig. 2.2.

First of all, we note that 100,000 simulations is not enough to accurately estimate the probabilities of extremely rare events (such as large S_W). The fact that one sequence out of 100,000 had $S_W = 8$ does not mean that $P(S_W = 8) \approx 1 \times 10^{-5}$. Therefore, we can only judge the accuracy of the heuristics by verifying the probabilities of events that happen many times during the course of our simulations. In spite of this, we notice that all four heuristics still agree for high values of k. This lends credence to their accuracy for these values, since for moderate values of k they all agree and are correct. It seems unlikely for all of them to agree on an incorrect result.

We can see that Naus' heuristic and our Tuned MC heuristic perform well for all k, but unlike Naus', ours has the useful properties of being extendable to multiple transcription factors in a straightforward manner.

2.3 Multiple transcription factors

The complexity of handling an unconstrained number of TFs is not much more than handling just two TFs, so we discuss this case first.

Once we allow two TFs, let's call them A and B, we can no longer define a "significant" module in terms of a single number k. There are many different combinations, even for just two factors, which we will need to consider. Let's say A occurs every 1000 bp and B occurs every 500 bp. We use the term a "simple definition" of a cluster to mean a criterion such as "at least X sites of A and

$R = 500$ (1 TFBS / 500 bp on average)

k	Sampled $\hat{P}(S_W = k)$	Naus	Poisson Clumping	Markov Chain	Tuned MC
1	0.0	6.36×10^{-18}	1.92×10^{-12}	3.07×10^{-14}	1.92×10^{-14}
2	1.00×10^{-4}	3.63×10^{-5}	8.33×10^{-5}	1.04×10^{-4}	4.04×10^{-5}
3	0.134	0.133	0.138	0.155	0.139
4	0.604	0.604	0.600	0.593	0.601
5	0.226	0.227	0.226	0.217	0.224
6	0.0322	0.0325	0.0324	0.0314	0.0323
7	3.47×10^{-3}	3.35×10^{-3}	3.35×10^{-3}	3.26×10^{-3}	3.36×10^{-3}
8	2.10×10^{-4}	2.88×10^{-4}	2.88×10^{-4}	2.82×10^{-4}	2.92×10^{-4}

$R = 1000$

k	Sampled $\hat{P}(S_W = k)$	Naus	Poisson Clumping	Markov Chain	Tuned MC
1	0.0	3.39×10^{-6}	1.29×10^{-5}	9.04×10^{-6}	2.71×10^{-6}
2	0.165	0.164	0.169	0.183	0.167
3	0.670	0.670	0.665	0.657	0.668
4	0.151	0.153	0.152	0.147	0.151
5	0.0127	0.0127	0.0127	0.0124	0.0127
6	6.90×10^{-4}	7.74×10^{-4}	7.73×10^{-4}	7.60×10^{-4}	7.74×10^{-4}
7	2.00×10^{-5}	3.86×10^{-5}	3.85×10^{-5}	3.79×10^{-5}	3.88×10^{-5}
8	1.00×10^{-5}	1.63×10^{-6}	1.63×10^{-6}	1.61×10^{-6}	1.66×10^{-6}

$R = 2000$

k	Sampled $\hat{P}(S_W = k)$	Naus	Poisson Clumping	Markov Chain	Tuned MC
1	0.0251	0.0253	0.0285	0.0306	0.0250
2	0.738	0.736	0.733	0.737	0.738
3	0.224	0.226	0.225	0.219	0.224
4	0.0125	0.0131	0.0131	0.0128	0.0130
5	3.70×10^{-4}	4.95×10^{-4}	4.94×10^{-4}	4.88×10^{-4}	4.93×10^{-4}
6	1.00×10^{-5}	1.48×10^{-5}	1.47×10^{-5}	1.46×10^{-5}	1.48×10^{-5}

Figure 2.2: Scan statistic approximation comparison

$\#A \setminus \#B$	0	1	2	3	4	5	6	7
0	1.0	1.00	1.00	1.000	0.866	0.263	0.0361	3.66×10^{-3}
1	1.00	1.00	1.000	0.962	0.464	0.0871	0.0106	1.04×10^{-3}
2	1.000	0.999	0.920	0.463	0.106	0.0155	1.76×10^{-3}	$\ddots$
3	0.836	0.611	0.279	0.0732	0.0129	1.75×10^{-3}	$\ddots$	$\ddots$
4	0.166	0.0878	0.0297	6.68×10^{-3}	1.11×10^{-3}	$\ddots$	$\ddots$	
5	0.0136	6.70×10^{-3}	2.13×10^{-3}	$\ddots$	$\ddots$	$\ddots$		
6	8.14×10^{-4}	$\ddots$	$\ddots$	$\ddots$				

Table 2.1: Fraction of 60kbp sequences containing simple clusters

at least Y sites of B" is a significant cluster. Table 2.1 shows for each simple definition of significant cluster, what fraction of sequences of length 60 kbp will contain such a cluster.

For example, this table shows that 96.2% of such sequences will have a cluster of at least 1 A site and 3 B sites in some window of 300 bp (this counts windows containing 1 A and 3 Bs, or 2 As and 3 Bs, or 1 A and 4 Bs, or 2 As and 4 Bs, and so on). As another example, 1.29% of sequences will have a cluster of at least 3 A sites and 4 B sites (from now on we will use the notation **3+4** to represent such a cluster).

For any given level of statistical significance, there are many simple cluster definitions whose probabilities are below it. Take for example 0.01. Clusters $0{+}n$ for $n \geq 7$ all have probability $< 10^{-2}$. Therefore it seems wise to only consider 0+7 (all the rest are redundant). Clusters $1{+}n$ for $n \geq 7$ have the same property, so again it seems wise to only consider 1+7. However, 1+7 is redundant when compared with 0+7 so we can ignore 1+7 as well. With similar reasoning, we find that a "minimal" set of simple definitions which all have probability less than 10^{-2} is 0+7, 2+6, 3+5, 4+3, 5+1, and 6+0. None of these is more general than any other. We call such a set of simple definitions a *complex cluster definition*.

We know the behavior of each of the simple definitions individually, but the behavior of this complex definition is not so clear. Its components are not independent, because some windows fulfill more than one simple definition (e.g., 3+7, which fulfills 0+7, 2+6 and 3+5). This means that the probability of observing an occurrence of the complex definition is not the same as the sum of observing each simple definition individually. We found each simple definition by starting with a threshold determined by statistical significance (10^{-2}) but the set as a whole has some greater probability. It seems that a more useful algorithm would bound the probability of the complex definition instead.

We propose a novel alternative statistic, which we call the *generalized scan statistic*. Rather than starting with a threshold in mind and using it to determine a cluster definition that fulfills it, we use this statistic to evaluate existing clusters. Like the standard scan statistic, we first define a random variable Y_i' for each window in long sequence. To handle multiple transcription factors, instead of being a single number or an ordered tuple representing the number of TFBSs within the window, Y_i' is the probability of observing the TFBSs in the window according to the Poisson distribution. For example, if the window starting at position i contains 5 As and 3 Bs,

$$
\begin{aligned}
Y_i' &= P(\geq 5 \text{ As})P(\geq 3 \text{ Bs}) \\
&= (1 - P(\leq 4 \text{ As}))(1 - P(\leq 2 \text{ Bs})) \\
&= (1 - \sum_{i=0}^{4} P(i \text{ As}))(1 - \sum_{i=0}^{2} P(i \text{ Bs})) \\
&= \left(1 - \sum_{i=0}^{4} e^{-\lambda_A} \frac{\lambda_A^i}{i!}\right)\left(1 - \sum_{i=0}^{2} e^{-\lambda_B} \frac{\lambda_B^i}{i!}\right)
\end{aligned}
$$

where $\lambda_A = p_A W$ and $\lambda_B = p_B W$, the expected numbers of A sites and B sites in a window of size W. Now we define concretely our generalized scan statistic

$$S'_W = \min_{1 \leq i \leq L-W+1} Y_i'$$

Rather than taking the maximum count, S'_W finds the minimum probability. For one transcription factor, this is equivalent to the standard scan statistic. But for more than one TF, this captures in a single number all the different combinations of TFBS counts in a simpler and more useful way than defining simple or complex definitions of clusters. We use this statistic to evaluate a given cluster by asking the question: what is the probability that a cluster of equal or even less likelihood is found in a sequence of similar length? We do this by evaluating the probability of the window containing the cluster, say p, and calculating $P(S'_W \leq p)$.

Our Markov chain heuristic (both tuned and naive) can handle this complex probability calculation by adding additional "goal" states. Every combination of TFBS counts which results in a probability of p or less is made into a goal state, so that $P(S'_W \leq p)$ is precisely the probability of reaching any one of those states in the Markov chain over the course of L transitions (where L is the length of the DNA sequence we are modeling).

2.4 Evaluation

We used this model to estimate the probability of observing a cluster of TFBSs of equal or lesser probability to that of the known *S. purpuratus otx* CRM [115]. This *cis*-regulatory module contains nine TFBSs (5 Gata, 2 Krox/Blimp1, and 3 Otx) in a 500 bp window. We empirically estimated the parameter of the Poisson distribution for each transcription factor by counting the number of occurrences of TFBSs in the surrounding DNA sequence, dividing by the length of the overall DNA sequence, and scaling this by the window size. Using these parameters, the probability of the window containing the CRM is 8.57×10^{-4}. Seeing such a window in 30 kbp of sequence, which is the length that we have been told by biologists is the amount they want to be able to search within, turns out to be 83%. We verified this by simulation. Because our initial definition of S'_W includes all combinations of TFBS counts that result in a $p \leq 8.57 \times 10^{-4}$, including combinations of only one or two TFs rather than all three, we tried restricting S'_W to consider only clusters containing all three. Still, about 69% of the 30 kbp sequences contained such clusters.

This proves that by considering only the TFBSs sequences and their clustering is not sufficient to recognize a *cis*-regulatory module. Otherwise, nearly every gene would contain such a CRM by chance alone. There must be additional requirements due to the protein-protein interactions that occur outside of the sequence. The purpose of the building *cis*-Lexicon is to enable to recognition of these requirements for future predictive algorithms.

Chapter 3

Prior Work

As described above, algorithmic approaches fail when attempting any of the three steps of cracking the *cis*-regulatory code. Predictive algorithms will only succeed when they utilize databases of accurate, experimentally-derived *cis*-regulatory architecture. Some databases do exist, such as ORegAnno, REDfly, TRANSFAC, and TRED [36, 39, 74, 47], but they accept TFBS annotations resulting from experiments such as DNase I footprinting, gel shifts, and ChIP-chip/seq. All of these techniques suffer from limited resolution—they report a region wherein a factor likely binds. ChIP (chromatin immunoprecipitation) methods in particular are known to be noisy; many of the putative regions it identifies will not in reality contain a binding site [26, 48]. [120], for example, reported that motifs recognized by PWMs were found within 100 bp of only "~60-80%" of their ChIP peaks. [9] searched the literature for retinoic acid binding sites and found 81 "tested and verified" experimentally, from which, upon close examination, at least 22 (>27%) appeared to be spurious. The results of techniques such as these cannot be taken as conclusive proof of the existence of regulatory regions or the lack thereof. They also cannot detect the regulatory function of the TFBSs or CRMs they do find. Details on the most popular *cis*-regulatory databases are as follows:

REDfly 1,354 of REDfly's 1,427 TFBSs (>94%) are validated by DNase I footprinting; only 6 are validated by *in vitro* reporter constructs and none by site-specific mutation/deletion. Focused on CRMs to the point where many do not contain any known TFBSs. Very few functional annotations.

ORegAnno Only 21 of ORegAnno's 14,361 TFBSs have site-directed mutagenesis as supporting evidence. The website seems neglected; the last posted news is from 2008 and attempting to register causes a server error.

TRANSFAC Commercial database; publicly accessible version dates back to 2005. No CRMs or regulatory function (the focus is on generating PWMs). Even the highest quality rating for TFBSs ("functionally confirmed") does not always satisfy our criteria (e.g., experiments using methylation interference)

TRED Only contains information on three species (human, mouse, and rat: all mammalian). The genome assemblies the annotations are based on are from 2003. The quality scale is too coarse—all experimentally-verified TFBSs are in one category, regardless of method

Other databases suffer from similar problems. For accurate conclusions to be drawn about the properties of regulatory regions and the distinctions between random clusters of binding site sequences as opposed to real regulatory modules, accurate data needs to be utilized. If any progress is to be made regarding predicting the actual function of CRMs (whether they activate or repress, and in what time and location), this information must be recorded for previously-known modules as well. [24] suggested that major reasons for the *cis*-regulatory logic code to not yet be understood is the lack of "sufficiently useful, discriminatory, and general target site databases" and that "the decisive importance of particular site and factor combinations has only been sporadically recorded."

Existing *cis*-regulatory databases usually do contain some type of annotation describing the "quality" of each element, this often does not give enough detail as to the type of experiment. Only experiments which pass what we call the Davidson Criteria yield results suitable for entry into the *cis*-Lexicon:

Davidson Criteria: Transcription factor binding sites must be functionally authenticated by site-specific mutagenesis, conducted *in vivo*, and followed by gene transfer and functional test [46]

Experiments fulfilling this criteria prove the causal links between genes in the gene regulatory network. They find the precise means by which one gene regulates another, which is through transcription factors binding to their sites in the *cis*-regulatory region of the target gene. Other techniques can only prove correlation or association.

Chapter 4

The *cis*-Lexicon: a database for *cis*-regulatory information

The *cis*-Lexicon is a database of *cis*-regulatory information. While other databases have been built in the past by various groups, these have all suffered the drawbacks discussed in the previous section.

The only place this information can be found is in the journal papers themselves, so for the last four years we have hired undergraduate biologists to read these papers, determine which meet our standards, and input the data into the *cis*-Lexicon via the *cis*-Browser. The *cis*-Lexicon now contains over 730 TFs binding over 2,300 sites in the regulatory regions of more than 570 target genes.

The *cis*-Lexicon contains the following types of annotations:

CRM coordinates It is not clear how to formulate a definition for the boundaries of a *cis*-regulatory module. Sequence conservation often extends beyond the functional binding sites, but this may only be due to evolutionary selection against large insertions or deletions within regulatory sequence [19]. Since it is unknown whether the precise boundaries are significant or not, the boundaries given in the paper are stored in the Lexicon along with a note as to how they were determined (whether by restriction sites, sequence conservation, or otherwise). The *general concept* of CRM boundaries is undoubtedly important, as the sites inside a CRM work together yet are functionally independent from sites in other modules.

TFBS coordinates Knowing the TFBS coordinates implies knowing both location and sequence. The location specifies the relationship with other sites which work in combination, as well as the distance to the transcription start site (which affects how the binding factor interacts with the transcription apparatus). The sequence of individual sites aids in defining models as to the general type of site a given TF binds to.

TFBS regulatory function TFBSs can be annotated with the regulatory functions that they fulfill: activation, repression, signal response, DNA looping, etc. The precise choices available

are terms from the *Cis*-Regulatory Ontology (CLO), which was designed by examining typical *cis*-regulatory analysis papers and distilling the various terms describing the same fundamental phenomena into a controlled vocabulary (see Fig, 4.1).

TFBS binding factors The binding factor (or factors, in the case of a complex) is specified for each site. In order to avoid the "Gene Naming Problem" (see [105]) where the precise identity of a gene is unknown because it is known by a set of names (sometimes overlapping with a set of names from a different gene), the NCBI GeneID is stored. This also allows for efficient algorithmic processing.

TF families Transcription factors exist in a hierarchy. There are multiple ways to classify them, and we chose a modified TRANSFAC system (see Fig. 4.2) to classify all of the TFs in the *cis*-Lexicon (both target genes and *cis*-regulatory inputs). For meta-analysis (see Section 4.3), for TFs which are not found in the *cis*-Lexicon enough times to generate reliable statistics, we plan to combine data of TFs within the same family.

Sequence conservation Occasionally papers note that *cis*-regulatory sequence, whether for individual binding sites or for entire modules, is conserved across species. Annotators can quickly record this knowledge in the *cis*-Lexicon. Sometimes the additional species may not have their whole genomes sequenced yet, such as opossum and elephant. Keeping this data in the Lexicon allows for the sequence to be annotated in those species when their full genomes are sequenced in the future.

Target gene function An open question concerning *cis*-regulatory regions is whether their architecture is fundamentally different between different types of genes—are the regulatory regions of transcription factor encoding genes different from those of housekeeping genes? To allow this type of analysis, annotators note in the Lexicon the type of each target gene.

Given our limited time and resources, we decided to focus on collecting the regulatory information of transcription factor-encoding genes in eight particular species only, for the current time: human, mouse, fruit fly, sea urchin, nematode, rat, chicken, and zebrafish, with the highest priority on the first five species. When completeness (see Section 4.2) of TF regulatory regions in these species is reached, then our focus will move to a new type of gene.

4.1 Implementation

The *cis*-Lexicon was originally stored in a set of XML files, one per target gene. This was the only file format supported by the Celera Genome Browser, since at Celera the browser used a private database whose details have not been released. Searching the Lexicon required the *cis*-Browser to open, read, and process every one of these files–and this was repeated for each individual search

Repression Indicates that mutating the TFBS increases gene expression or produces ectopic expression. Repressors may act "long range," when the repression effect may target more than one enhancer, or "short range," when repression affects only neighboring activators [38, 21]. The function of repression applies in cases where the repressors interact with the basal transcription apparatus either directly or indirectly [76].

Activation Indicates that mutation decreases gene expression. An activator TFBS may act over a large genomic distance or short. See [57] for further discussion of some of the many ways a transcription factor can accomplish activation.

Signal response Indicates that the transcription factor has been shown to be activated by a ligand such as a hormone (phosphorylation is not included) [10].

DNA looping Indicates that the binding factor is involved in a protein-protein interaction with another binding factor some distance away that causes the DNA to form one or more loops. This looping brings distant regulatory elements closer to each other and to the basal transcription apparatus [119].

Booster Indicates that the TFBS does not increase gene expression on its own but can augment activation by other TFBSs.

Input into AND logic Indicates that the TFBS can activate gene expression only when two or more cooperating TFBSs are bound [44, 45].

Input into OR logic Indicates that the TFBS can activate gene expression when either or both of two or more cooperating TFBSs are bound [44, 45].

Linker Indicates that a TFBS is responsible for communicating between CRMs (such as the CB2, CG1, or P sites in modules A and B of endo16 [117])—mutating the TFBS prevents the functions of the independent modules from combining.

Driver Indicates that this TFBS is the primary determining factor of gene expression. The binding factor appears only in certain developmental situations and thus is the key input for directing gene expression. TFBSs that are not drivers usually bind ubiquitous factors [102].

Communication with BTA (basal transcription apparatus) Indicates that the sites are directly involved with interactions with the BTA (many sites are only indirectly involved—they use other sites as mediators)

Insulator Indicates that the TFBS causes *cis*-regulatory elements to be kept separate from one another. Insulators can separate the *cis*-regulatory elements of different genes as well as act as a barricade to keep active segments of DNA free of histones and remain active [112].

Figure 4.1: The *Cis*-Regulatory Ontology (CRO), a controlled vocabulary for describing *cis*-regulatory function [46]

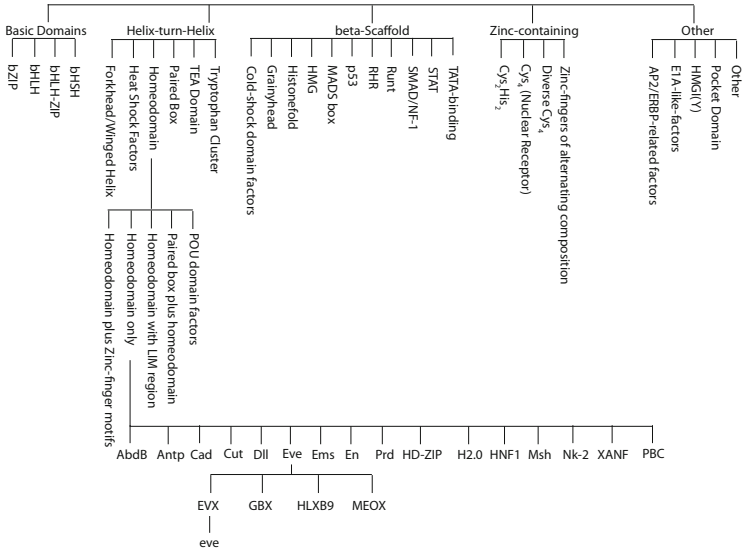


Figure 4.2: Transcription factory hierarchy

request (some simple queries used cached indexes, but many could not). An interim measure to allow certain restricted searches (e.g., list all the genes in the lexicon; ask whether a given gene is in the lexicon) was a simple *ad hoc* index created by an external tool. We designed a database schema for storing this information in a way which allows data to be imported/exported to/from the XML format without a loss of data. This schema helps to prevent errors and allows for the regulatory information to be queried efficiently, utilizing relational database features such as indexes as foreign keys . By indexing the NCBI GeneID field of gene records, for example, searching for genes by GeneID immediately becomes fast. Foreign keys are utilized to ensure data integrity. For example, if the bound factor for a binding site is recorded as being gene 373400, then the database will ensure that gene 373400 is already present in the *cis*-Lexicon, or else reject the annotation.

We implemented the *cis*-Lexicon using Apache Derby, an open source relational database. Since Apache Derby is implemented entirely in Java, the *cis*-Browser remains entirely cross-platform. Derby can be run either in embedded mode, where the database is stored and accessed locally, or as a network client, where the database is stored remotely and accessed via a server. This allows the *cis*-Lexicon to be packaged with the *cis*-Browser for ease of access or to be stored in one central location so users of the *cis*-Browser around the world will see an updated database from the moment a change is made.

4.2 Completeness

Completeness of the *cis*-Lexicon is critical in order for correct conclusions to be drawn. There are two kinds of completeness:

1. Biological completeness, where we know all *cis*-regulatory information
2. Literature completeness, where we have captured all information available in the literature

The first type of completeness is unfortunately not possible, as this information is simply unavailable. Therefore we aim to be as close to literature completeness as possible, and this is what we refer to when we use the term "completeness" without qualification. We use several methods to judge completeness, including

- Transcription factor counts per species
- Inspection by domain experts
- Consulting literature reviews
- Literature searches (CLOSE; see Section 6)
- Other regulatory databases

A pessimistic estimate of literature completeness can be given by evaluating biological completeness. This is possible because the set of all transcription factors is known for species whose genomes have been sequenced. There are standard tools for recognizing which genes encode transcription factors, even if the genes have not been studied experimentally.

We have carried out routine reviews of the Lexicon by domain experts, who verify that genes are classified correctly and that the most well-known modules are in place. While one might expect that the best-understood modules are the most likely to have their literature found and entered into the Lexicon, actually the reverse is often true: the most popular genes tend to have so many papers discussing them that it can be difficult to find the original articles where the *cis*-regulatory analysis was performed. Recognizing them is especially difficult with automated methods like CLOSE (see Section 6). For this reason, completeness tests based on these genes are less biased that one would expect.

Many of the papers which discuss but do not actually perform *cis*-regulatory analysis cite several papers which do carry it out. We used many of these to test the Lexicon as well.

CLOSE is a key method for testing completeness. We routinely take samples of 100-500 papers from CLOSE results and examine them to see how many discuss *cis*-regulatory analysis of genes not yet in the Lexicon. Recently only 1-2% of the papers have contained novel information. This is a strong sign that there are few genes remaining to be found.

Our annotators have combed through existing databases such as REDfly and TRANSFAC to look for genes not yet in the Lexicon. This is a tedious process because even when the evidence given in those databases does not meet our criteria, they check whether newer literature has been published on those genes which does.

All of these methods give evidence to the *cis*-Lexicon being nearly literature complete. See Section 6.5 for details.

4.3 Analysis

We performed several preliminary analyses of the data in the *cis*-Lexicon in order to extract properties of *cis*-regulatory regions that distinguish them from chance clusters of transcription factor binding sites (that, due to the size of the genome, occur thousands of times in the genome of any complex species). Without a clear understanding of what distinguishes true regulatory regions, past work has involved looking for clusters of TFBSs or conserved sequence, which are neither sufficient nor necessary.

4.3.1 TF Co-occurrence

Different transcription factors whose sites occur within the same *cis*-regulatory module belong to the same *regulatory state*. The regulatory state of a cell at a given time is the set of transcription factors that are expressed in the cell at that time. It is called the regulatory state since future gene regulation in the cell is determined almost entirely by which transcription factors are present. Cells that adopt different fates (for example, becoming part of the eye versus becoming part of the heart) do this due to a difference in regulatory state. Not all states possible in theory occur in practice. If there are 20,000 genes in a species, for example, this doesn't mean that $2^{20,000}$ regulatory states are involved in the development of an organism of that species. In reality a much smaller set of states occur, determined largely by the possible ways that the transcription factor proteins (and their co-factors) can interact with each other to influence transcription.

To discover transcription factors that are part of the same regulatory state, we searched the *cis*-Lexicon for transcription factors that bind within the same CRMs. An idea of the complexity of this information alone can be seen by inspecting the Human, mouse, and fruit fly connectivity graphs, seen in Fig. 4.3. These graphs, which also represent the "View from the Genome" described in [24], contain a vertex for each gene and draw a directed edge between two vertices if the first gene regulates the second. These graphs are not gene regulatory networks (GRNs), but rather the union of multiple GRNs, showing the relationships between genes at various times and locations in the development of an organism. False conclusions could be drawn if these graphs were interpreted like GRNs. For example, if there is an edge from Gene A to Gene B and another edge from Gene B to Gene C, this does not mean necessarily that the expression of Gene A affects the expression of Gene C. It may be that the two edges in the graph are from different times or locations, and at no time does Gene A influence Gene C. But connectivity graphs are useful for visualizing the complexity of the regulatory genome and the completeness of the *cis*-Lexicon.

The most common pairs of transcription factors often included SP1, SP3, and their homologs, which are ubiquitously expressed in mammalian cells and are known to regulate genes involved in almost all types of cellular processes by interacting with a variety of proteins [62]. Binding sites for these factors are not informative, so we do not present or discuss them. The most common pairs not involving SP1 or SP3 can be seen in Table 4.1.

The number of binding sites of each transcription factor was not considered—e.g., if a CRM

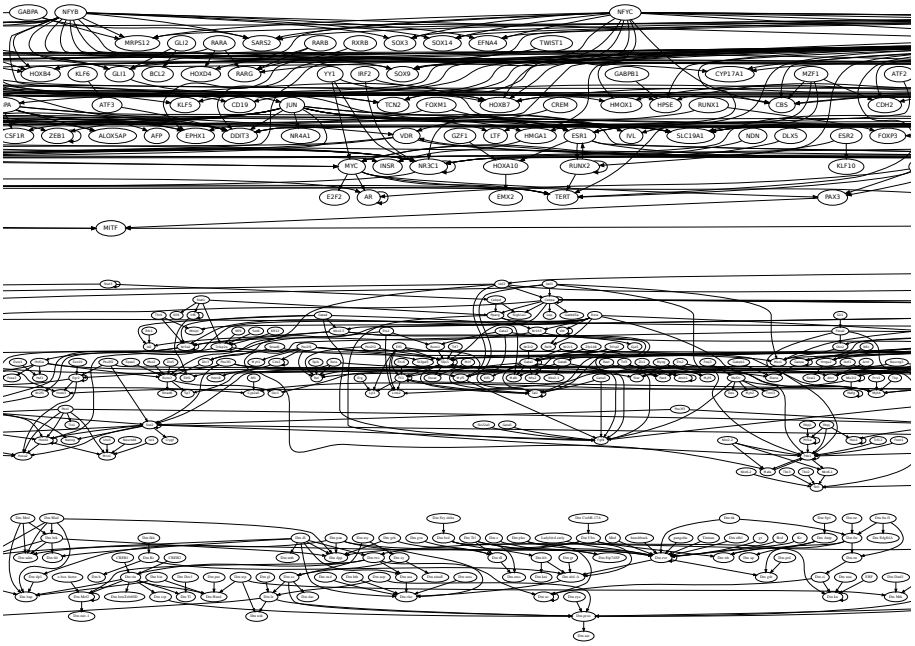


Figure 4.3: Excerpts from the transcription factor connectivity graphs for the human, mouse, and fruit fly genomes

TF 1	TF 2	# CRMs	# Target Genes
Mm Pou5f1 (18999)	Sox2 (20674)	4	4
Dm Mad (33529)	Med (43725)	3	3
Mm Hoxb1 (15407)	Pbx1 (18514)	3	3
Dm exd (32567)	hth (41273)	3	3
Hs HNF4A (3172)	HNF1A (6927)	3	3
Dm brk (31665)	Mad (33529)	3	3
Mm Nkx2-2 (18088)	Pdx1 (18609)	3	3
Hs CEBPB (1051)	DBP (1628)	3	3
Mm Ptf1a (19213)	Rbpj (19664)	3	2
Rn Usf2 (81817)	Usf1 (83586)	2	2
Hs USF1 (7391)	USF2 (7392)	2	2
Hs RXRA (6256)	NR2F2 (7026)	2	2
Hs NFYA/B/C (4800)	USF1 (7391)	2	2
Dm dl (35047)	twi (37655)	2	2
Gg PAX6 (395943)	SOX2 (396105)	2	2

Table 4.1: Pairs of transcription factors observed to bind within the same *cis*-regulatory modules. Species' names are abbreviated (*M. musculus* is *Mm*, etc), and the unique NCBI GeneID for each gene is given in parentheses.

contained 2 TFBSs for Mad and 3 for Med, this was counted as a single occurrence of the pair Mad-Med. We examined both the number of CRMs that a pair was found in as well as the number of unique target genes, although these were generally the same. The only exception was murine Ptf1a and Rbpj, which was found regulating two genes but in three CRMs (1 CRM of Pdx1 and 2 CRMs of Ptf1a itself). There were many more examples of pairs found exactly twice (approximately 30 more), but they are not presented here to save space (since they are not strong evidence of transcription factor cooperation). However, we note that several well known examples of transcription factors that combine to form regulatory complexes currently have only two examples in the *cis*-Lexicon (e.g., dl-twi [118] and PAX6-SOX2 [50]). This does not mean that the *cis*-Lexicon is missing information—for those two examples, the complexes were recognized by high-throughput methods and careful experimental analysis of a single gene, respectively. It's also interesting to note that transcription factors that often bind within the same CRM do not necessarily work together—it is possible that their binding sites overlap so that they *compete* for occupancy in order to carry out their regulatory function, such as brk and mad [51].

4.3.2 Inter-TFBS Spacing

Not all transcription factors part of the same regulatory state necessarily interact with each other directly. For example, the five transcription factors which bind to the ten sites of Module A of endo16 do not all interact. Transcription factors which directly interact with each other are said to be *cooperative*. One example of such a pair already known to work together is Dorsal and Twist or Snail [118]. TFs which bind cooperatively tend to bind a specific distance apart from each other. This allows the proteins to interact without any need for DNA looping (see [119] for a review). This can only occur when the binding sites have the specific, correct distance between them. If the sites are too far apart, the proteins cannot come into contact with each other without some sort of looping; if the sites are too close, the transcription factors cannot bind simultaneously. Consistency of distance is strong evidence that two TFs do in fact work cooperatively. Even if two factors often bind within the same CRM, if the distance between them appears random, it is not clear whether they interact via DNA looping or whether they do not interact at all.

The analysis we discussed in the previous section permitted the binding sites of the two factors to be situated anywhere within the CRM. To detect cooperativity, we scanned the *cis*-Lexicon to find examples of pairs of transcription factors whose sites are consistently a nearly constant distance apart. Since binding sites are input into the *cis*-Lexicon exactly as they are given in the literature, their boundaries are not consistent. For example, one paper might give a 4 bp binding site while another will present an 8 bp binding site. Different binding sites may represent different parts of the overall binding motif. To allow for these types of differences, we simply searched for pairs of binding sites for which the distance between them was less than 20 bp. This is the maximum typical distance that allows neighboring bound proteins to interact. The results are summarized in Table 4.2.

Unlike the TF co-occurrence analysis above, in which there were very many pairs that were found regulating exactly two target genes, the set of results here was much smaller (as to be expected).

TF 1	TF 2	# Occurrences	# Target Genes
Mm Pou5f1 (18999)	Sox2 (20674)	4	4
Dm Mad (33529)	Med (43725)	3	3
Dm exd (32567)	hth (41273)	3	3
Mm Ptf1a (19213)	Rbpj (19664)	3	2
Mm Pbx1 (18514)	Pknox1 (18771)	3	2
Mm Hoxb1 (15407)	Pknox1 (18771)	3	2
Gg PAX6 (395943)	SOX2 (396105)	2	2
Hs PDX1 (3651)	HNF1A (6927)	2	2
Hs PDX1 (3651)	NEUROD1 (4760)	2	2
Hs HNF4A (3172)	HNF1A (6927)	2	2
Hs GABPA (2551)	NFYA/B/C (4802)	2	2
Mm Pou2f1 (18986)	Sox2 (20674)	2	2
Mm Pou2f1 (18986)	Sfpi1 (20375)	2	2
Mm Gata1 (14460)	Tcfcp2 (21422)	2	2

Table 4.2: Pairs of transcription factors whose sites are observed to bind nearby each other within the same *cis*-regulatory modules. Species' names are abbreviated (*M. musculus* is *Mm*, etc), and the unique NCBI GeneID for each gene is given in parentheses.

All pairs found regulating two or more target genes are shown in Table 4.2.

4.3.3 TFBS Multiplicity

In some CRMs, a single transcription factor binds at many different sites, such as Kni in the stripes 3+7 regulatory module of the *D. melanogaster* gene *eve* (12 binding sites) and Gata in the Otx15 module of the *S. purpuratus* gene *otx* (5 binding sites). Other TFs bind only once within a CRM, such as Slp in *DmDll* and Brn1/2/4 in *SpEndo16* [24]. We searched the *cis*-Lexicon to recognize the transcription factors which tend to bind at many sites within a single CRM, only once, or have no clear pattern. Some of the results are shown in Table 4.3. Most TFs appeared to bind only once or twice per CRM.

4.3.4 Interspecies Analysis

The above three analyses treated every gene as a unique entity. This is the most conservative and safest type of analysis, but not the most powerful. More correlations can be recognized if the same genes in multiple species are grouped together. For example, the transcription factors HNF1A and HNF4A in human have been observed to bind near each other twice in human, and the TFs Hnf1a and Hnf4a have been observed to bind near each other once in rat. Separately, these observations are not very significant. When pooled together to yield three observations of the same pair of transcription factors, the evidence is much stronger.

Determining genes in different species to be the "same" is nontrivial. Generally, when two genes are said to be the same, what is meant is that the genes are "orthologous": they are descend evolutionarily from a common ancestor. Seeing the names HNF1A and Hnf1a might lead one to

Transcription factor	# CRMs	# Genes	TFBSs/CRM	Category
Hs JUN (3725)	16	15	2 TFBSs (2 CRMs); 1 TFBS (14 CRMs)	Single
Mm Rxra (20181)	11	10	2 (2 CRMs); 1 (9 CRMs)	Single
Mm Gata1 (14460)	11	8	2 (6 CRMs); 1 (5 CRMs)	Single
Hs HNF4A (3172)	10	9	2 (1 CRM); 1 (9 CRMs)	Single
Mm Sfpi1 (20375)	8	7	3 (3 CRMs); 2 (1 CRM); 1 (4 CRMs)	Varies
Dm dl (35047)	7	7	3-4 (4 CRMs); 2 (3 CRMs)	Multiple
Dm Ubx (42034)	6	5	4-12 (3 CRMs); 1-2 (3 CRMs)	Varies
Dm Su(H) (34881)	6	6	3-7 (5 CRMs); 1 (1 CRM)	Multiple
Dm Mad (33529)	5	5	9 (1 CRM); 4 (1 CRM); 1-2 (3 CRMs)	Varies
Dm srp (41944)	4	3	5 (1 CRM); 3 (3 CRMs)	Multiple

Table 4.3: Categorization of various transcription factors according to the number of binding sites found in single CRMs

expect that genes that are the same will have the same name. This is often not the case, as in the trio of genes *Rbpj* (in mouse), *Su(H)* (in *Drosophila*), and *lag-1* (in *C. elegans*) (this gene is also called *RBPJ* in human and *Su(h)* in the sea urchin). Many genes have interesting scientific histories behind them and are often named after the phenotypes they cause when mutated, which varies from species to species even when the basic function of the gene is the same. Oftentimes a journal paper will give synonyms for the gene or genes under study, such as "CEH-22/tinman/Nkx2.5", "Wnt/MAPK", "POP-1/TCF", and "SYS-1/beta-catenin" mentioned in [55].

Databases that attempt to record orthology relationships, such as NCBI Homologene [90]and InParanoid [82], are based on automated sequence comparisons that cannot take into account the functional relationships between related genes. Thus, one will often find mentioned in papers that *tinman* and *Nkx2.5* are synonyms, but one will fail to find this relationship in databases. This is due to the fact that there is a family of related transcription factors, and if one considers only the sequence, one will judge another factor in *Drosophila* other than *tinman* to be more closely related to *Nkx2.5*, and *tinman* to be more closely related to something other than *Nkx2.5*. The reason that these two genes are considered synonyms in spite of the difference in sequence is due to their shared function: they are both involved in the development of the heart. There is no perfect solution to this problem other than carefully recording all synonyms reported in the literature. While we have been doing this as part of the *cis*-Lexicon annotations, this has not been a focus, and our records

		# CRMs	# Target Genes	# Species
Hnf4a	Hnf1a	5	5	3
Pou5f1	Sox2	5	5	2
Nr1h3	Rxra	4	4	2
Mad/Smad1	Med/Smad4	4	4	2
exd/Pbx1	hth/Meis1	4	4	2
Hoxb1	Pbx1	4	4	2
Srf	Creb1	3	3	3
Srf	Egr1	3	3	3
Srf	Elk1	3	3	2
Gata4	Nkx2-5	3	3	2
Pbx1	Pknox1	3	3	2
Rxra	Nr2f2	3	3	2
Hoxb1	Pknox1	3	3	2
Hnf4a	Nr2f2	3	3	2

Table 4.4: Interspecies TF Co-occurrence analysis

		# CRMs	# Target Genes	# Species
hth/Meis1	exd/Pbx1	4	4	2
Su(H)/Rbpj	da/Tcf12	4	2	2
Hnf4a	Hnf1a	3	3	2
Srf	Creb1	3	2	3
Gata4	Nkx2-5	4	2	2
Tcf3	Hnf1a	2	2	2
Smad2/3/4	Fos/Jun	2	2	2
Gata1	Fli1	2	2	2

Table 4.5: Interspecies inter-TFBS spacing analysis

are not complete enough to be relied upon. The papers we use to annotate the *cis*-Lexicon are not consistent in citing synonyms since they are not necessary. For this reason, we have relied on NCBI HomoloGene as an imperfect but reasonably-complete source of interspecies synonyms.

Results from interspecies TF co-occurrence analysis can be seen in Table 4.4. This table only shows pairs of transcription factors seen in at least three CRMs and in multiple species. Results from interspecies inter-TFBS spacing analysis are presented in Table 4.5. Again, this table only shows pairs of transcription factors found in multiple species.

More surprising results came from the interspecies TFBS multiplicity analysis. Transcription factors known to bind multiply in one species tended to bind singly in other species, such as Su(H) in *Drosophila*, which binds multiply (as noted above), while the mammalian homolog Rbpj tends to bind singly in mouse and human (one CRM in mouse contains three sites for Rbpj while the other five known CRMs in mouse and human contain only one site). Similarly, while Dl usually binds multiply as mentioned earlier, its mammalian homolog Rela has only a single site in eleven of the twelve CRMs discovered in human, mouse, and rat. The other transcription factor found to bind multiply, srp, does not have homologs binding in the *cis*-Lexicon. We did not observe any

transcription factor to consistently bind multiply across different species.

Chapter 5

The *cis*-Browser: A genome browser for *cis*-regulatory analysis

The CYRENE *cis*-Browser is a genome browser tailored for *cis*-regulatory annotation and investigation. It is the sole means through which *cis*-regulatory information is input into the *cis*-Lexicon, and the chief method of visualizing and interacting with the Lexicon's data. The *cis*-Browser began as a branch of the Celera Genome Browser, which is freely available as open source at `http://sourceforge.net/projects/celeragb/`. The features of the original Celera Genome Browser focused on viewing and annotating gene transcripts, so many new capabilities were added to address the new focus.

The *cis*-Regulatory Browser Problem Create a single environment which allows biologists to add, edit, and interact with rich *cis*-regulatory information

The *cis*-Browser has been used for experiments and cited by the Davidson Lab at Caltech. Presentations on the *cis*-Browser have been given at the last three Developmental Biology of the Sea Urchin conferences (XVIII, XIX, and XX); at the most recent conference, the release of *cis*-Browser was publicly announced. The Browser (containing the gene *endo16*[1]) is currently available for download at the Istrail Lab web site: `http://www.brown.edu/Research/Istrail_Lab/`.

5.1 Implementation

Firstly, support for *cis*-regulatory modules (CRMs) and transcription factor binding sites (TFBSs) was added. Each of these new types of genomic features possesses several unique properties and associated information. Unlike gene transcripts, whose borders are determined solely by their exons, the boundaries of CRMs can extend beyond the known binding sites contained inside (e.g., if evidenced by sequence conservation). It is often known whether or not whole CRMs or individual binding

[1] since the *cis*-Lexicon has not yet been released

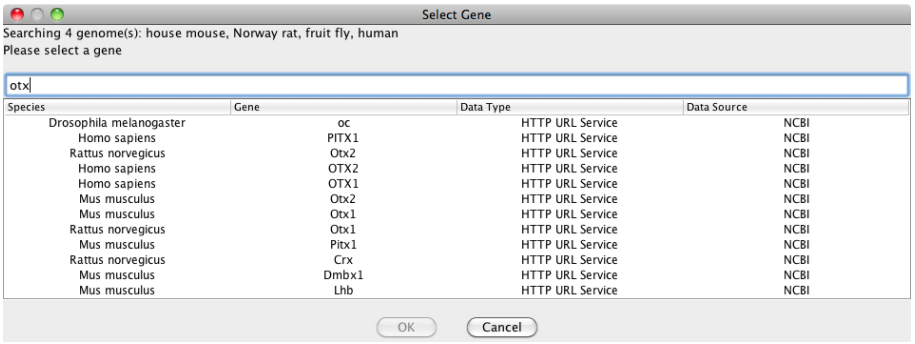


Figure 5.1: Searching NCBI Entrez Gene via the *cis*-Browser

sites are conserved across species—this information can be added and viewed via the *cis*-Browser. Each TFBS has a specific factor (or, occasionally, a family of related factors) which binds there. The NCBI GeneID for this factor (or its name, if unknown) and its effect on gene expression can be annotated and viewed in the *cis*-Browser. We added the ability to color TFBSs differently according to their binding factor, so that the architecture of *cis*-regulatory regions can be visualized easily. Also, we added on-screen labels to genes, CRMs, and TFBSs so that their identities are visible at a glance.

The focus of the CYRENE *cis*-Browser is on annotations that are supplemental to genes that are already known, rather than discovering transcripts. Therefore, instead of requiring annotators to input the genes themselves, the capability was added to download the genes directly from NCBI. Within the *cis*-Browser application, the user can search for genes (see Fig. 5.1) in the same manner they would use the NCBI Entrez Gene web site. Upon selecting a gene from the results, the genomic sequence of the region is downloaded and all of the gene's transcripts and exons are displayed, automatically. This capability is considered significant by the current maintainer of the Celera Genome Browser, who plans to port the functionality back.

In the Celera Genome Browser, properties of genomic entities could be of two types: (1) plain text (e.g., names) or (2) a choice from a list of options (e.g., evidence type: *cis*-mutation, footprinting, etc). Properties can be nested, so that a single (parent) property can contain inside it several additional (child) properties. For *cis*-regulatory annotation, we required accurate recording of complex properties. Firstly, we needed to support properties containing multiple interdependent parts: for example, when annotating the factor that binds at a certain site, we must keep track of the name of the factor, its NCBI GeneID, and any synonyms mentioned in the literature. Secondly, we needed to support multiple values for a single property: multiple synonyms, multiple *cis*-regulatory functions, and conservation across multiple species.

For properties with multiple parts, we created rich dialog boxes that ensure the user enters correct information. It would be tedious to demand that the user flip back and forth between the *cis*-Browser and the NCBI Entrez Gene web site to look up GeneIDs for each binding factor. It would be error-prone to require the user to manually type the factor names and GeneIDs, especially when

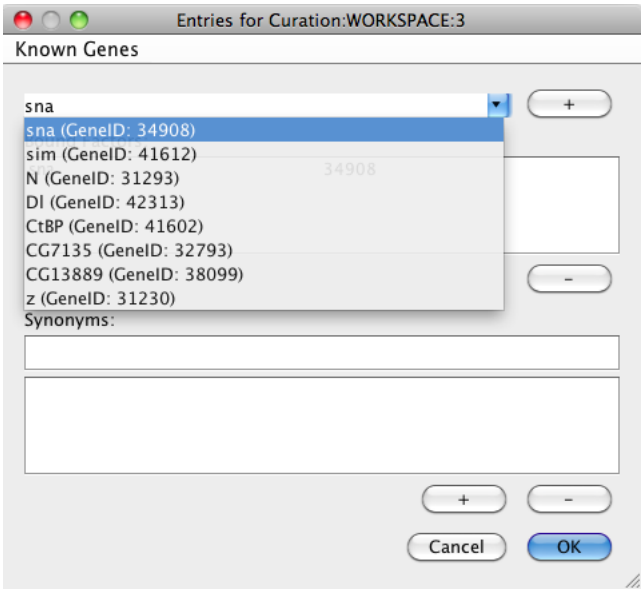


Figure 5.2: Searching NCBI Entrez Gene for a bound factor

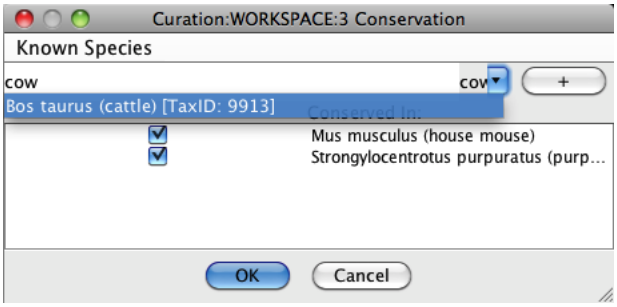


Figure 5.3: Searching NCBI Taxonomy for a species to note sequence conservation

the same factor binds at several sites in the regulatory region of a single target gene. Therefore the *cis*-Browser contains a special dialog for annotating the factor that binds to each site. It allows the user to search the Entrez Gene site from within the browser, automatically restricting the search to the species being annotated. If the same factor binds at multiple sites, for the second and later sites, the user may select the gene from a menu rather than re-entering the information and performing another search (see Fig. 5.2). There are similar windows for annotating conserved species (which searches NCBI for the correct scientific names and NCBI tax ID), and *cis*-regulatory functions (which ensures that the *Cis*-Regulatory Ontology is followed in naming the regulatory function and verifies that PMIDs are typed correctly)—see Figs. 5.3 and 5.4.

As for supporting properties with multiple values, three different mechanisms were tested before a final implementation was decided on. The first two involved extensions to the core data source interface of the Browser and changes to the XML file format. After significant debate we decided that we preferred compatibility with the Celera Genome Browser (which, being open source, is still

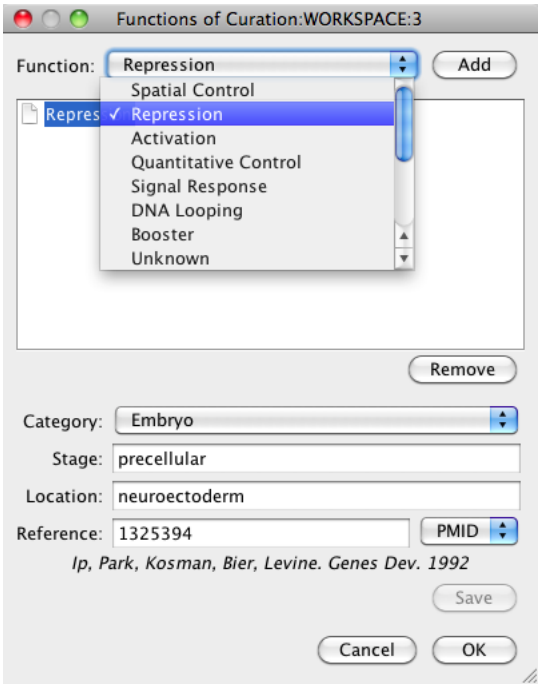


Figure 5.4: Annotating the regulatory function of a transcription factor binding site

under development and still used) and other applications which may handle the GAME format.

Therefore a simple convention for encoding multiple-valued attributes as properties was decided upon: a set of values would be represented by a single property whose children contain the information but are given unique names. For example:

```
<property name="cisreg_functions" value="4">
  <property editable="false" name="cis_functions0" value="active"/>
  <property editable="false" name="cis_functions1" value="required_by"/>
  <property editable="false" name="cis_functions2" value="and_input"/>
  <property editable="false" name="cis_functions3" value="signal"/>
</property>
```

(This excerpt hides additional details of the annotation, such as time, location, and citation for each function).

At Celera, the Genome Browser was part of a three-tiered application communicating with an application server to access a database. The Genome Browser supported loading genomic features from files, but this was meant to supplement the database (with, for example, output from bioinformatics tools), not replace it. The interface to the *cis*-Lexicon is implemented as an entirely new data source, accessing an Apache Derby database (for details see Section 4.1). Currently the database is embedded within the Browser, but support for connecting to a centralized remote database is already implemented and can be activated by changing a single line in a configuration file.

We also added support for the GFF (General Feature Format) file format, a standard format for exchanging information on genomic features. Many bioinformatics tools including Apollo, Argo, Chado, CMap, GBrowse, IGV, and others support this format natively. All genomic features in the *cis*-Browser can be exported in GFF for processing with other tools. This allows information

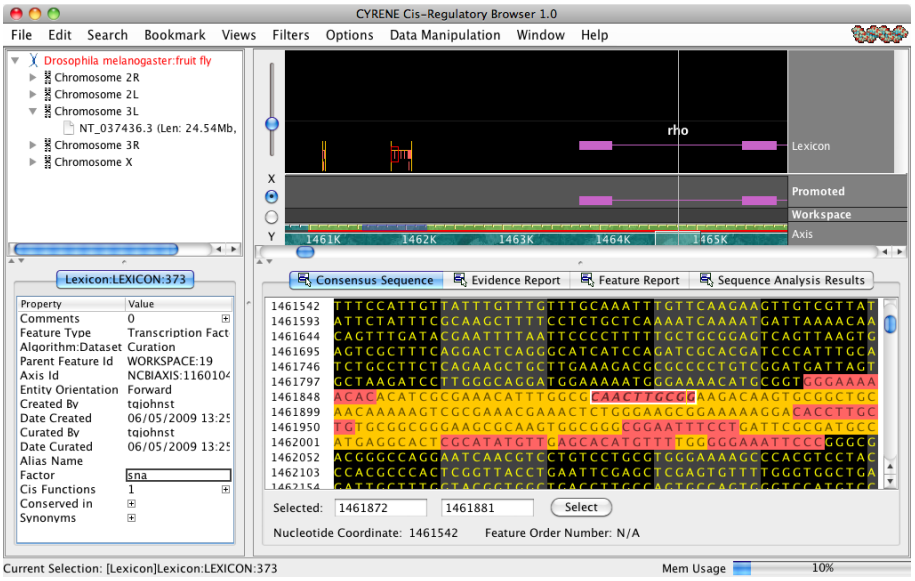


Figure 5.5: The *cis*-Browser interface

from the *cis*-Lexicon to be easily utilized as part of a workflow involving tools from various sources without needing to be explicitly designed to work together. Support for loading GFF files is planned for a future release.

Additional features were added to support next generation sequencing and read mapping as well (see Section 5.5).

5.2 The *cis*-Browser Interface

The CYRENE *cis*-Browser interface shares the same organization as the original Celera Genome Browser. The *cis*-Browser application window is split into four regions: (clockwise from the top-left) the Outline View, the Annotation View, the Subview Container, and the Property Inspector View (see Fig. 5.5). The Outline View (top-left) displays in a hierarchical tree format the species, chromosomes, and sequences loaded by the *cis*-Browser and ready for analysis. The Annotation View displays the locations of genomic features (e.g. transcripts, CRMs, etc.) on the sequence currently being examined. The Subview Container shows the user a set of views specific to the currently-selected feature. The Property Inspector View displays the properties of the currently-selected feature in textual form.

The Annotation View (top-right in Fig. 5.5) allows real-time zooming from a chromosome-wide view down to the individual nucleotide level. The colors are configurable, but Fig. 5.5 shows the exons of the gene *rho* in purple, the CRM boundaries in yellow, and the TFBSs in red. The user can select a genomic feature by clicking on it, after which information specific to the feature is visible in the Subview Container and the Property Inspector View. The Annotation View displays genomic features in tiers, which are horizontal rows that group features according to their source, so that

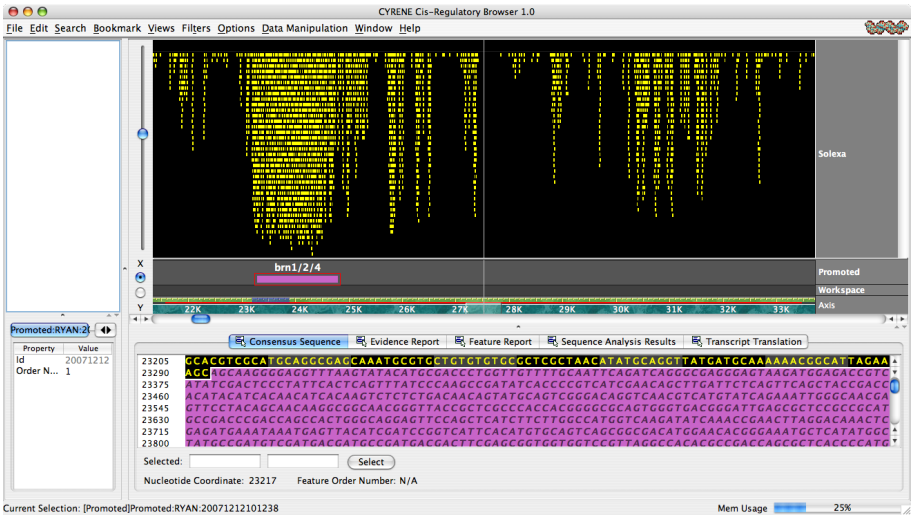


Figure 5.6: *cis*-Browser genomic feature tiers

information from multiple sources is not intermixed and confused. See Fig. 5.6 for an example where Solexa reads are grouped separately from genomic transcripts. One special tier in the Annotation View is the workspace. The workspace is the tier containing genomic features currently under editing. Features loaded from data sources such as XML files or the *cis*-Lexicon are considered immutable, so a copy must be made before an annotator may modify it.

Each type of genomic feature has particular traits that distinguish it from other types. For examine, transcripts are translated into proteins, and BLAST hits are the result of comparisons between different sequences. Therefore, a view for viewing the translation of DNA codons into amino acids is relevant only for transcripts, while a view for examining the differences between the current sequence and a sequence which was searched against it is relevant only for BLAST hits. The Subview Container (bottom-right in Fig. 5.5) is the location for views such as these. When a feature is selected in the Annotation View, only the views relevant to that type of feature are shown in the Subview Container. Only one such view is shown at a time, to maximize the visible area. The rest are shown as tabs that user may click on to switch to that view.

One subview of critical importance is the Consensus Sequence View. It displays the sequence of the selected feature and the surrounding region. This view is also used to specify the location of new features. The user simply clicks and drags to select sequence in the same way as selecting text in a word processor or web page. Right-clicking shows a menu with options to create a transcript, CRM, or TFBS. The seqFinder, built into the Consensus Sequence View, is a tool for quickly locating the exact coordinates of a sequence contained in a published paper. Given a region of sequence to search within (e.g. a gene and its flanking sequence), the seqFinder allows the user to type in the minimum amount of nucleotides to uniquely find the paper's sequence. For each letter the user types, the seqFinder reports whether the sequence typed so far is found more than once (i.e. multiple ambiguous matches, so more input is necessary to determine which is correct), exactly once (i.e., a

Figure 5.7: The seqFinder tool of the *cis*-Browser

perfect match; no more typing needed), or never (i.e., a typo or possibly a true mismatch between the paper's sequence and the reference genome). By typing the minimum amount to find the beginning and end of the paper's sequence, the precise coordinates are located quickly and accurately. See Fig. 5.7 for a typical use of seqFinder.

Every genomic feature has certain properties associated with it, such as name, NCBI accession number, date of curation, and so on. The Property Inspector View (bottom-left in Fig. 5.5) displays these properties as a two-column table, where the name of each property is on the left and the value is on the right. Properties can be edited by double-clicking the current value. If the value is a simple string, then it may be edited in place. If the value is more complex, such as binding factors, then a dialog box will open. Property changes, when they affect how the feature appears in the Annotation View, are reflected in real time. If the name of a gene or CRM is modified, for example, then the new name appears immediately in the Annotation View.

5.3 *Cis*-Regulatory Features

5.3.1 *Cis*-Regulatory Structure Diagramming

We added the ability to produce publication-quality graphics of the *cis*-regulatory structures that biologists enter. Examples of typical diagrams from existing papers can be seen in Fig. 5.8. This feature, aside from the obvious benefit of making these diagrams easier to generate and more consistent across publications (which is especially helpful for biomedical text mining techniques that take into account image information [97]), also automatically distributes the task of entering new *cis*-regulatory modules to the labs that actually perform the work. Once the information is entered to generate a figure for a paper, after the paper is reviewed and published, the information can be easily transmitted and included within the *cis*-Lexicon. In fact, we hope that in the near future journals will require *cis*-regulatory information associated with a paper to be deposited in a public database, such as the *cis*-Lexicon, before publishing. Many journals already require this for various other types of data, such as microarray data and nucleotide or protein sequences. One example of

a diagram generated by the *cis*-Browser is shown in Figure 5.9.

5.3.2 Navigation by *cis*-Regulatory Relationships

We added to the *cis*-Browser the ability to navigate the genome via the *cis*-regulatory relationships between genes. One can right click on a gene and see the transcription factors that regulate it, as well as (when the target gene is a transcription factor itself) the downstream genes that it regulates. Right-clicking on a binding site shows the precise factors that bind at that site. Each gene shown in the menu is in fact a submenu, giving the option to either move to and load that gene, or to recursively view its regulators and targets. An example is shown in Figure 5.10, where a binding site in the *Drosophila eve* mesodermal enhancer is selected. The menu shows that *pan* binds at the site, and that *pan* is known to regulate *slp1*, *eve, dpp*, and *oc*. Finally, *oc* is shown to be regulated by *ci*, *pan*, *oc* (itself), and *bcd*. Clicking the "Open" choice in each menu would load the respective gene in the *cis*-Browser.

5.3.3 Gene Regulatory Network View

We added an embedded gene regulatory network (GRN) view, an alternative view of the genome that is as important as the structural view. [70] introduced BioTapestry, an application for modeling gene regulatory networks, including both simple visualization as well as the comparison of experimental data with hypothesized regulatory links. Among other uses, BioTapestry has been used to model the *S. purpuratus* endomesoderm network, mouse ventral neural tube specification, the T-cell gene regulatory network, and a network containing most of the genes of *Halobacterium salinarum*.

We collaborated with the author of BioTapestry to embed a read-only version of the application into the *cis*-Browser, as can be seen in Figure 5.11. When the user selects a gene in the Annotation View, that gene becomes selected in the GRN. When a gene is selected in the GRN View, a dialog box appears asking whether to load that gene in the *cis*-Browser. This is more user-friendly than automatically loading the gene, since (unlike in the GRN View) only one gene can be viewed at a time and it takes a non-trivial amount of time to load a gene. Future work will also involve making the *cis*-Browser and BioTapestry programs communicate, most likely through the Gaggle, a framework for exchanging data between independently developed biology software tools already supported by BioTapestry [94].

5.4 Annotating the *cis*-Lexicon

For each genomic feature entered, the annotators first input the coordinates by locating them with the seqFinder. The relevant properties such as names, binding factors, *cis*-regulatory functions, and sequence conservation are set via the Property Inspector View. The Annotation View makes for quick sanity checks—are the binding sites located upstream, downstream, or within introns of the regulated gene, as is usually the case? Are the CRMs of a reasonable size?

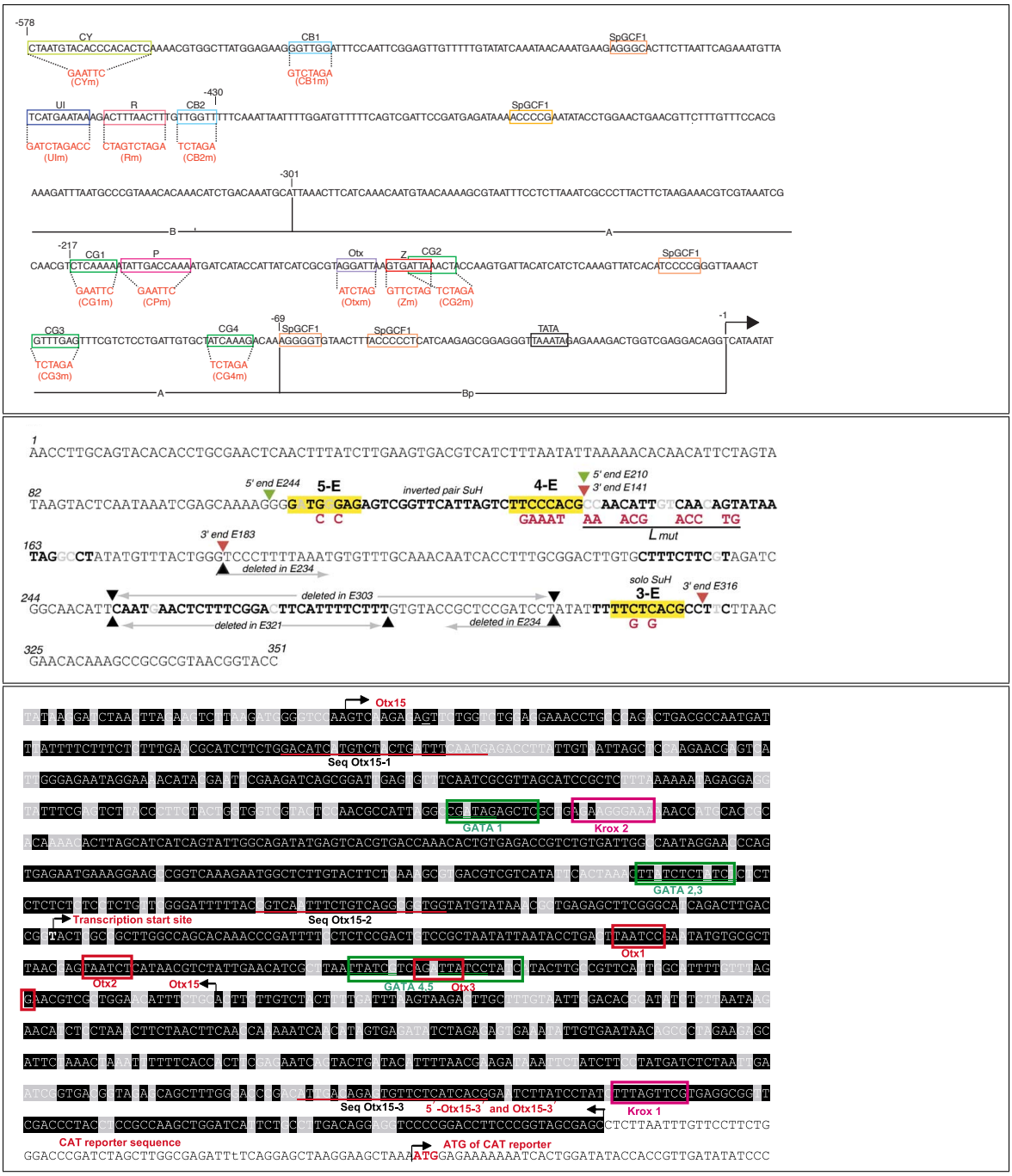


Figure 5.8: Examples of *cis*-regulatory structure diagrams (from [117, 85, 115])

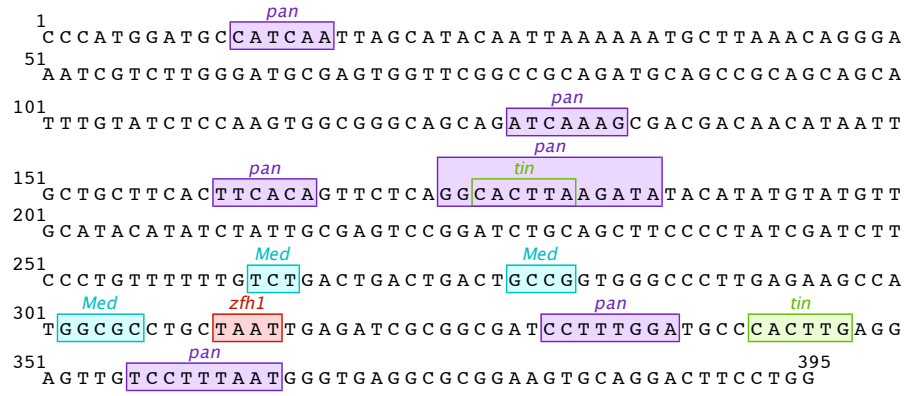


Figure 5.9: *Cis*-regulatory structure diagram of the *Drosophila eve* mesodermal enhancer [52] generated by the *cis*-Browser

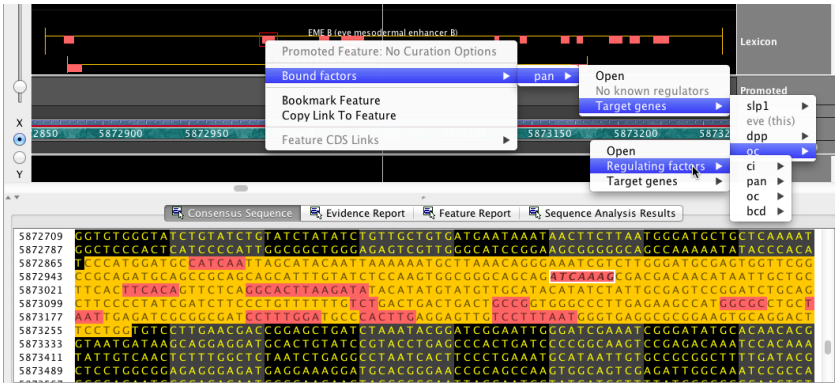


Figure 5.10: Navigating *cis*-regulatory relationships in the *cis*-Lexicon via the *cis*-Browser

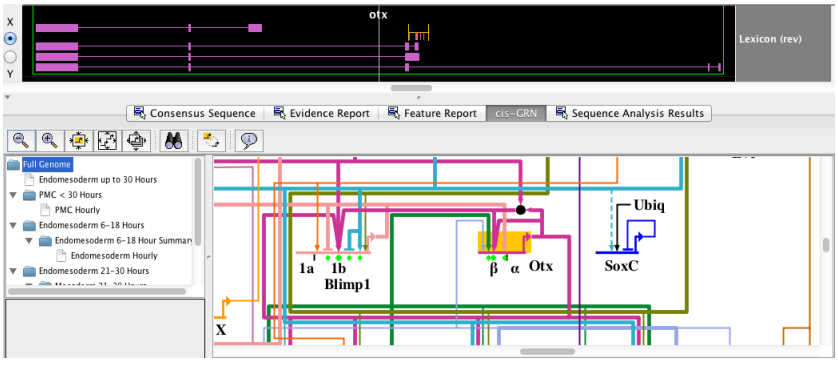


Figure 5.11: *Otx* in the *S. purpuratus* endomesoderm gene regulatory network, visualized in the *cis*-Browser. The gene was manually selected in the Annotation View, causing it to be automatically selected in the GRN View

The annotators' work is saved as XML files in the GAME format, rather than directly inputted into the *cis*-Lexicon. This allows for easy backup and sharing of past work, as well as preventing cluttering the database with half-finished or unchecked annotations. A special software tool is required to move the annotations from these intermediate files into the *cis*-Lexicon. By forcing the use of intermediate files and preventing unauthorized annotators from modifying the *cis*-Lexicon directly, the database can be kept at a strict high level of quality. An experienced annotator can verify the work of a new trainee before it is entered into the *cis*-Lexicon.

5.5 SMAPPER: An algorithm for short read mapping

As mentioned above in Section 1.1.3, a common method for finding regulatory sequence is to look for noncoding sequence conserved between two species of the proper evolutionary distance (often 50 million years apart). It is rare that the sequence of two species of the correct evolutionary distance is available, so acquiring this sequence (through genome sequencing and assembly) becomes a time-consuming and expensive process.

In order to make this process easier and less costly, we developed a specialized tool to enable the Davidson Lab at Caltech to map inexpensive Solexa reads to existing genomic scaffolds or BACs. They had previously used FamilyRelations [15], an application developed in-house, for interspecies comparisons but this tool could only compare full sequences (not reads). Our goal was to map 25-bp Solexa reads from *L. variegatus* to the *S. purpuratus* genome, allowing up to 4 mismatches. This permits the discovery of *cis*-regulatory regions in *S. purpuratus* without a pre-existing *L. variegatus* genome assembly and without attempting to assemble the short reads (which is error prone).

The Short Read Mapping Problem Given a set of millions of 25-character sequences (next-generation sequencing reads) and several longer sequences (BACs) each a few hundred thousand characters long: for each read, find every substring in each long sequence which is equal to that read with up to 4 mismatches

This is a very specific case of the approximate string matching with Hamming distance problem. Initially we used RMAP [100], but we required extra flexibility such as outputting all of the locations each read mapped to instead of ignoring reads which mapped to more than one location[2]. Also, existing read mappers are not optimized for such high tolerance for mismatches. Read mappers are generally used for mapping reads to the genome of the same species as the reads, where differences are either polymorphisms in the genome (relatively rare) or errors in the sequencing process (which are automatically corrected by obtaining more reads). In such cases, with short reads, only one or two mismatches are typically permitted efficiently. Because we were mapping from one species to another, more mismatches must be allowed, because the species are millions of years apart in evolution and more differences will definitely exist that need to be tolerated.

[2]Newer versions of RMAP do have this feature [101]

In order to search the genomic sequence for approximate matches, we used a gapped seed index (see [16] for a detailed introduction). This is a generalization of a standard approximate string matching technique which looks for k-mers shared between the query and subject sequences. For example, if a 25-bp read matches a region in the genome with four mismatches, there are four facts known about this region:

1. The read is not guaranteed to contain even one 6-mer identical with the region.
2. The read must contain at least one 5-mer identical with the region
3. The read must contain at least six 4-mers identical with the region
4. The read must contain at least eleven 3-mers identical with the region
5. The read must contain at least sixteen 2-mers identical with the region

To see this, imagine the worst case scenario for four mismatches for each k-mer length. For 6-mers, the worst case scenario occurs when every five matches is followed by a mismatch, preventing a shared 6-mer from occurring. Visually, `-----x-----x-----x-----x-`. With this arrangement of mismatches, there is not even one shared 6-mer. For 5-mers, the worst case is when every four matches is followed by a mismatch. This can only happen four times (since we are allowing four mismatches), so one shared 5-mer must always exist—visually, `----x----x----x----x-----`. For 4-mers, the worst case looks like this: `---x---x---x---x---------`. At least six (overlapping) shared 4-mers must be present. Six is called the *threshold* for 4-mers, and one was the threshold for 5-mers. The 3-mer and 2-mer cases are analogous. These observations were formalized in the q-gram lemma of [49], which gave a simple formula for computing these counts. Recognizing that shared k-mers are always present motivates the creation of a k-mer index for the genome, which allows a quick lookup for each read to find all the locations in the genome where k-mers are shared. If, for example, six 4-mers are shared between a read and a certain location, then a full string comparison is performed to verify whether the location is in fact a true match for the read. This is necessary because the existence of six 4-mers is necessary but not sufficient; it is not a guarantee that the location is a valid match for the read. For example, `xxxxxxxx---------xxxxxxxx` contains sixteen mismatches yet still contains six shared 4-mers.

Because shared k-mers can overlap, many shared k-mers are generated by a single stretch of contiguous shared sequence, while in the "worst case scenario" described above, the mismatches prevent more than one shared k-mer from being generated at a time. It appears that permitting mismatches that are spread out requires us to set the threshold so low in order to detect them that many false positives are generated. If, for example, we required the mismatches to be more clustered, there would be more contiguous matching sequence and the threshold could be set higher. For example, if mismatches must occur in pairs, the worst case for 4-mers would be `---xx---xx---------------`, with at least twelve 4-mers always present—that is, having a threshold of twelve. Or, if mismatches must occur with at most two matches in between, `---x--x---x--x-----------`, giving a threshold of 8. This type of approach would result in a heuristic—it would make searches more efficient at the

	1 hit	2 hits	3 hits
4 mismatches	`#--#-#----#--#-#`	`##---#-##`	`#--#---------##`
5 mismatches	`##-###`	`#-#---##`	`#--------#-------#`

Table 5.1: Gapped seeds used in SMAPPER

cost of missing some real matches. Our collaborators, concerned with the accuracy of probabilistic mappers which map a high proportion but not all of the reads (e.g. [71, 64]), requested that our mapper be guaranteed to map every read possible. Probabilistic mappers are generally based on assumptions about polymorphisms within a genome or about the types of errors that can occur during sequencing, and these assumptions do not apply to interspecies comparisons.

An alternative approach is to use *gapped q-grams*, rather than k-mers. Rather than indexing contiguous short sequences, we index and search using gapped sequences according to some predefined *shape*. One example shape is `#-##`, which means use the first, third and fourth letters, ignoring the second. Applying the shape `#-##` to the sequence GATTACA yields the gapped q-grams GTT, ATA, TAC and TCA. Past work has shown that gapped q-grams perform better due to reducing dependencies between characters and increasing the *coverage* of the sequence caused by overlapping q-grams [16]. One can see that an arrangement of mismatches that prevents a shared 3-mer from occurring may not prevent a gapped 3-gram from being shared (for example, `--x--` is still matched by the shape `#-##`)

We used the method of [28] to determine the optimal gapped seed shape according to our criteria: the most specific (i.e., generating the fewest false positives) of all seeds sensitive enough to detect every true mapping. See Table 5.1 for the seeds found by this method. We then wrote the SMAPPER algorithm, which works in two phases: it first generates a hash table of sequences found by applying the gapped seed at every position in the genomic sequence. We only used seed shapes with a threshold of one, to reduce memory usage—we do not need to keep track of the number of hits in order to determine whether the threshold has been reached. Secondly, for each read, it checks each sequence generated by the seed against the hash table. Only where there are hits (i.e., a gapped q-gram from the read is found in the genome at this location) is a full base-by-base comparison performed. If the base-by-base comparison confirms that the read matches the genome within the mismatch tolerance specified, then the mapping is recorded. Unlike most other tools which report only the best mapping for a read (the location with the fewest mismatches) or entirely discard reads which have multiple mappings, SMAPPER returns all possible mappings. This is critical in interspecies read mapping because of the expected differences between the two species; the "best" mapping may not be the correct one, so it is important to see all possibilities.

Most read-mapping software works by analyzing the set of reads first, and then scanning the genome to look for matches. This makes sense because typically the set of reads is smaller than the genome. In our case, however, we do not need to scan the entire genome: when we map reads from the region around one *L. variegatus* gene, we only need to look in the *S. purpuratus* genome around its homologous gene. Therefore, our genomic sequence tends to be much smaller than the

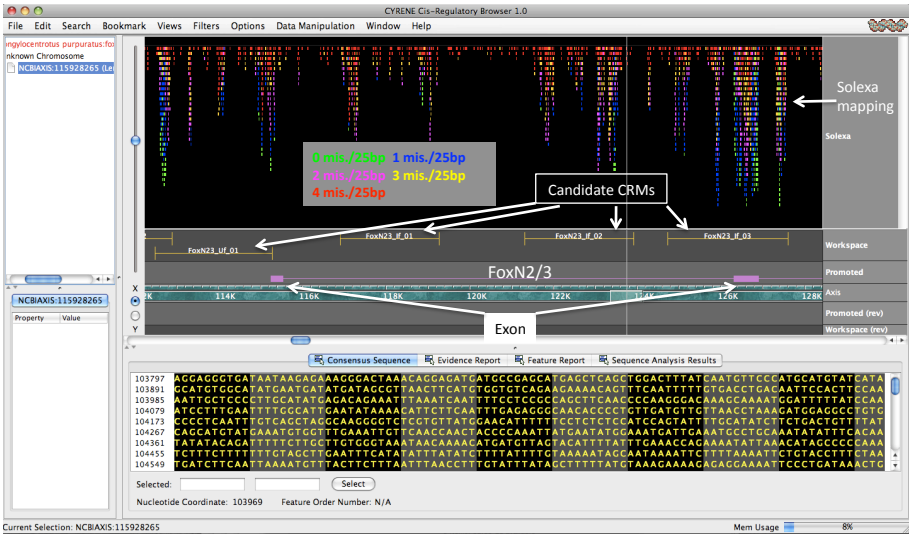


Figure 5.12: Figure S1 from [77], showing reads mapped by SMAPPER and visualized in the *cis*-Browser

set of reads. That is why the first phase of SMAPPER builds a hash table describing the genomic sequence rather than the reads.

Being a filtration technique, the gapped seed approach does not affect the time complexity of the SMAPPER algorithm. The seed index prevents the entire genome from being searched but the number of false positives which are discarded by the base-by-base verification is still proportional to the genome length. The time complexity is still $O(nml)$ where n is the length of the genome, m is the number of reads, and l is the length of a read. However, the index significantly decreases the constant scaling factor hidden by big-O notation.

This program was used and cited in a 2010 PNAS paper with the Davidson Lab [77]. Reads mapped by SMAPPER were visualized in the *cis*-Browser. Clusters of reads in non-repetitive regions were judged by biologists to delineate possible conserved regulatory regions, which were then extracted and tested for function. Fig. 5.12 contains a figure from this paper illustrating how SMAPPER and the *cis*-Browser were used. Gene exons are shown in purple on their own row. The reads appear in one of five colors, depending on the number of mismatches required to map the read to that location: green (no mismatches), blue (1 mismatch), magenta (2), yellow (3), or red (4). The user can easily switch between different tolerance levels to see only the reads mapped with the given number of mismatches or less (i.e., by default all mapped reads are shown, but reads with at most 3, 2, 1, or 0 can be shown by choosing the threshold from a menu). The biologist who carried out the work in the paper used the CRM annotation tools of the *cis*-Browser to mark the regions that appeared to be CRMs, followed by exporting these annotations and carrying out functional testing in the wetlab.

A few years later we developed a new version of the Solexa mapper, based on the SlideSort algorithm [99]. It is more flexible in terms of read length (supporting reads of any length), but

more significantly it can handle not only single nucleotide differences but insertions and deletions as well. We also developed a specialized version of the new tool specialized for comparing two large contigs or BAC sequences. This in essence performs the same function as FamilyRelations [15] but is much faster. In order to analyze and in particular visualize the results of the new algorithm, we have been using Atavist, an interactive comparative-genomics tool designed and implemented by our collaborator Russell Turner [108]. This tool is not yet publicly available, but is intended to be open-sourced in the future. We have contributed new features to his code base, including support for mouse wheel zooming (as we already added to the Celera Genome Browser when creating the CYRENE *cis*-Browser), textual labels on genomic features, and customized colors for individual features. This last feature is key for another use we have for Atavist, which is comparing the regulatory regions of homologous or co-regulated genes. By coloring each binding site according to its bound factor, it is easy to see at a glance the similarities and differences between regulatory regions. Using the *cis*-Lexicon together with Atavist allows such a view to be created automatically.

Since the original SMAPPER was written, several new approaches and tools have been published, including Bowtie, BWA, RazerS, SHRiMP, SOAP and ZOOM [56, 61, 111, 88, 23, 66, 67, 68]. Some some recent tools still use a gapped seed approach [68, 111, 23], while many mappers are now based on the FM-index datastructure [56, 60, 67]. The FM-index is based on the Burrows-Wheeler Transform used in data compression, recognizing its relationship with the suffix array datastructure [27, 17, 72]. The FM-index allows a highly compressed version of the genome to be searched for exact matches efficiently. Seed index approaches tend to need a large amount of RAM to fit the entire genome in memory at one time or require splitting the genome into pieces (SHRiMP2's index of the human genome is 48 GB, and RazerS's experiments are reported to have been carried out on a machine with 64 GB of RAM [23, 111]). Tools based on the FM-index often fit the entire genome index in the memory of a typical desktop machine, such as 2 GB [56]. The main downside of the FM-index is its lack of support for approximate string matching. Only the presence or absence of a specific sequence of characters can be queried. There are two approaches to overcome this: either use backtracking methods when traversing the index to account for differences between a read and the reference genome, or look up ungapped seeds rather than the entire read. The first technique is used by Bowtie and BWA (with different backtracking heuristics—unrestricted backtracking is too slow), while the second is used by SOAP2.

We are experimenting with approaches for combining the FM-index with gapped seeds, in order to obtain both low memory usage and speed along with sensitivity in the presence of mismatches. The Burrows-Wheeler Transform (BWT) involves a step where (simplistically) all suffixes of the string to be compressed are sorted lexicographically, like in a suffix array. This is what makes the FM-index an excellent datastructure—all exact occurrences of a pattern in the original string end up in a contiguous range in this sorted array. Currently we are investigating changes to this sorting step: if we vary of the order of the columns in the lexicographical sort, the final result will group together gapped seeds. See Fig. 5.13 for an example using the string CABCARCUB. Fig. 5.13a shows a typical lexicographical sort of the columns, while Fig. 5.13b shows a variation where the

(a)	(b)
ABCARCUB	ABCARCUB
ARCUB	ARCUB
B	B
BCARCUB	BCARCUB
CABCARCUB	**CABCARCUB**
CARCUB	**CUB**
CUB	CARCUB
RCUB	RCUB
UB	UB

Figure 5.13: Suffix sorting variations of the string CABCARCUB (see text)

third column is compared before the second column. This groups CABCARCUB and CUB together because they both have C as the first letter and B as the third letter. CARCUB, even though it shares the prefix CA with CABCARCUB, follows CUB because the third letter of CARCUB is R.This sort in fact will bring into a contiguous range all suffixes whose first and third letters are equal, regardless of the second letter. However, this sorting does not interact well with the BWT, because it removes the Last-First Mapping property necessary to recover the original text in a straightforward manner [27]. This does not mean that an efficient inverse transform is not possible; other variations of the BWT that lose this property, such as the Sort Transform [92], have been shown to have efficient inverse transforms very similar to that of the BWT [80].

We are also planning to extend SMAPPER to handle RNA-seq data, having begun collaborations with Joel Smith at the MBL and Marta Gomez-Chiarri at URI. RNA-seq is the process of sequencing the transcriptome, the set of mRNA molecules transcribed from the genome at a particular time and location within an organism. Mapping RNA-seq reads is especially complex due to RNA splicing. When RNA is first transcribed from the DNA of the genome, it contains sequence that is not meant to be protein-coding. These regions are called introns. Through a variety of mechanisms, these introns are cut out of the sequence, resulting in mRNA. This mRNA therefore contains sequence coming from disjoint areas of the genome. While an mRNA molecule is long and its sequence should map uniquely to one region in the genome, the reads generated by the RNA-seq process are short. It is difficult to map the reads that span sequence where the introns were cut out. Existing tools like TopHat and MapSplice attempt to solve this problem [107, 109], but we are formulating our own approaches based on SMAPPER which we will evaluate. The most promising approach currently is to use different gapped seeds applied to specific parts of the read, similar to ZOOM [68]. This allows us to detect efficiently when one end of the read maps to one location and the other end maps to a different but nearby location on the genome.

Chapter 6

CLOSE: the *cis*-Lexicon Ontology Search Engine

With the *cis*-Browser and *cis*-Lexicon efficiently allowing annotators to input and store *cis*-regulatory information, the key bottleneck becomes finding relevant journal articles for the annotators to read. One of the key goals of the *cis*-Lexicon project is to be complete: to contain (nearly) all of the information available in the literature (see Section 4.2). Some help is found through the other regulatory databases mentioned above by following their citations. More help is found through surveys, reviews, and books, such as [24]. But the other databases are incomplete, and even books only claim to give explanatory examples, not to be comprehensive indexes. Therefore, we began to develop the *Cis*-Lexicon Ontology Search Engine (CLOSE) to search the literature to automatically detect relevant papers (which we call *cis*-regulatory papers). We call this the *Cis-Regulatory Paper Problem*:

***Cis*-Regulatory Paper Problem** Find most journal publications with information relevant to the *cis*-Lexicon while minimizing false positives

Finding most is necessary for Lexicon completeness, while minimizing false positives is necessary to allow annotators to actually examine every paper. As no approach is perfect, any technique will require a (possibly arbitrary) balance between these two constraints. We have found the F-measure, the weighted harmonic mean of precision and recall, to be a useful measure of success [73]. By varying the weighting, we can emphasize precision over recall or vice versa. This begs the question: how do we calculate precision or recall? For precision, we make the naive assumption that all novel papers returned are irrelevant. This is actually a reasonable approximation because we expect only (roughly) one thousand out of a million papers to be relevant. With this assumption, increasing precision means minimizing the size of the returned set. To estimate recall, we simply see how many papers from our training set (which we call CYRENE papers) are recovered.

We began a collaboration with Hagit Shatkay, a leading expert on biomedical information retrieval and data mining [96, 95, 113, 98, 4]. Hagit was a member of the team that took first place

in Task 1 of the KDD Cup 2002, where the goal was to recognize whether papers met Flybase gene-expression curation criteria [86]. She visited our lab and we discussed approaches to information retrieval that would be most relevant for our specific problem—not detecting a theme or topic, as most systems do, but determining whether papers fit our experimental criteria. Lab members also met with Matt Lease, then a PhD Candidate at Brown University with experience in biomedical parsing [59], to discuss natural language processing strategies.

Until now, CLOSE has progressed through two stages, with a proposed third stage: automatic, human-designed, and hybrid.

6.1 Automatic CLOSE Algorithms

Automatic CLOSE used machine learning algorithms to distinguish between relevant and irrelevant articles. Our first approach, based on a desire to get results as quickly as possible, utilized the "Related citations" feature of PubMed to avoid the need to code our own algorithm. When a user views an abstract in PubMed, most articles will have a "Related citations" section on the upper right corner of the screen. This section lists a number of articles which an automated algorithm previously determined to be related to the article the user is currently reading [30]. An article may have none, a few, or even thousands of related citations. These citations can be accessed programmatically through NCBI's Entrez Programming Utilities web service [29]. We used this service to find the related citations for each of the papers our annotators had already found to contain information for the *cis*-Lexicon (CYRENE papers). If one citation was judged by NCBI to be related to several CYRENE papers, we took this as strong evidence that it was also relevant. This algorithm worked reasonably well, but was very limited. While many of the papers it found were indeed relevant (i.e., it had high precision), most of our CYRENE papers were not among the results, so we knew that it was not adequate in terms of completeness (i.e., it had low recall). We recognized that we needed to write our own algorithm from the ground up to be able to run on all of PubMed.

Before writing our own algorithm, we needed to determine what papers we would be analyzing. We examined the papers our annotators had used thus far and noted the journals that those papers were found in. We then used these journals only for future analysis, utilizing all papers published from 1992 onwards. Few papers before 1992 used experimental methods that fit our criteria, so including earlier years would likely give us mostly false positives.

Our second approach for automatic CLOSE comprised implementing a Naive Bayes classifier that operated on unigram and bigram tokens (since recognizing actual biology terms is very difficult, and sometimes even subjective) composed of words from the title and abstract.

To see what terms were bringing the highest-ranked results to the top, we looked at the five highest-scoring terms of each of the top 1,000 scoring papers, and observed which terms appeared the most often. The results can be seen in Table 6.1. The algorithm recognized certain terms that were used consistently but were not quite representative of what we were looking for. Experimental methods which do not meet our criteria but are often performed alongside of them (such as supershift

Term	Analysis
transcription	This is a useful term, since we are studying transcriptional regulation
Sp1	Could bias results; a specific transcription factor (see text)
enhancer	Useful, but weak evidence since many irrelevant papers also use this term ("enhancer" is used in different ways by different biologists)
Sp3	Could bias results; a specific transcription factor (see text)
supershift	Weak evidence, because it refers to an experimental method (supershift assaying) we do not accept
minimal promoter	Potentially useful; but "promoter", like "enhancer," is used to mean many related things
proximal promoter	"
basal promoter	"
promoter activity	"

Table 6.1: Terms found in the papers ranked highest by the Naive Bayes classifier

assay) were given a strong weighting, and significance was also attached to the names of common regulatory factors such as Sp1. While these are indeed *correlated* with relevant papers, they would bring in a great deal of noise and raise to the top of the ranking papers which would not fit our criteria or papers that discuss genes regulated by factors which are not specific enough to be useful (e.g., Sp1 indiscriminately boosts the expression of many target genes but is never the deciding factor in determining the binary fact of whether a gene is expressed or not). Approaches using variations of the standard TF-IDF (term frequency–inverse document frequency [73]) scoring system failed to improve the results.

This algorithm was confounded by the inconsistent terminology used by biologists—the key concepts that needed to be recognized to detect a relevant paper could be written in numerous different forms. For example, the a transcription factor binding site may be described as a binding motif, *cis*-element, *cis*-binding element, target site, or one of many other terms. The variation of names led the algorithm to judge each individual name to be a rare term that did not have any importance. Unfortunately, these terms are exactly the ones that a biologist would recognize and use to judge an abstract.

6.2 Expert-Designed CLOSE Algorithm

To overcome the problem of term variation, we aimed to implement a system of rules that mimic how a biologist identifies a relevant paper. Through discussions with a biologist collaborator, we developed four rules based on lists of related terms that only a biologist would know, such as the various names for TFBSs and the steps involved in the different experimental techniques that we accept. We applied these rules to a set of 607 papers known to be relevant to determine which papers

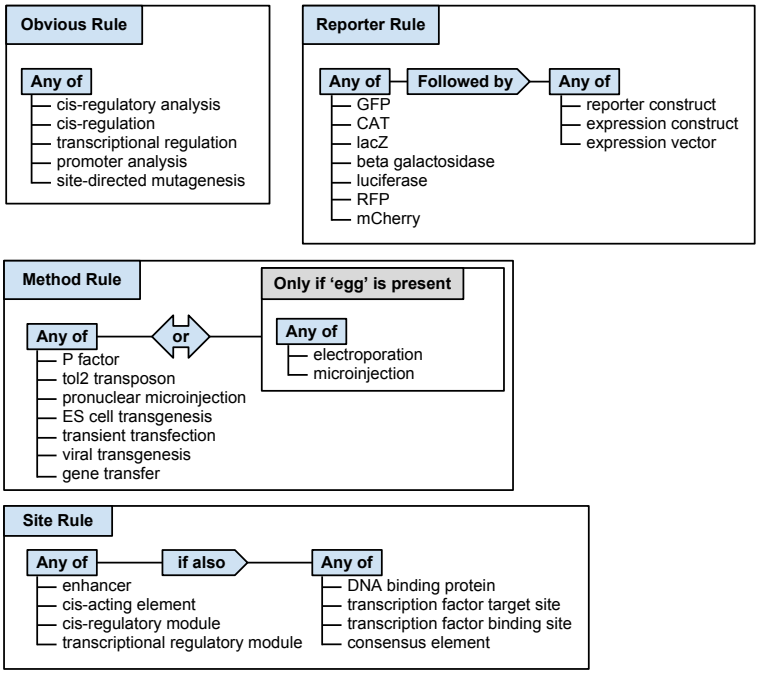


Figure 6.1: The four original Davidson Rules, as suggested by Eric Davidson (later extended with more terms and an additional rule: the Action Rule, matching terms such as "activation", "repression", "boosting", etc)

	0 rules	**$\geq$ 1 rules**	**$\geq$ 2 rules**	**$\geq$ 3 rules**	**$\geq$ 4 rules**	**All 5 rules**
# PubMed	706,630 (75.6%)	227,976 (24.4%)	119,969 (12.8%)	42,375 (4.53%)	11,011 (1.18%)	2,283 (0.24%)
# CYRENE	46 (7.6%)	561 (92.4%)	546 (90.0%)	439 (72.3%)	223 (36.7%)	65 (10.7%)

Table 6.2: Results from applying five Davidson Rules to PubMed titles/abstracts

were missed. We expanded the lists of synonyms and added an additional rule. We implemented these rules efficiently using a Rete-like algorithm [31]. Applying these five rules, which we called the Davidson Rules (see Fig. 6.1), resulted in each paper matching some subset of these rules. Few papers ($\sim$10%) matched all five. Therefore, we still needed a criterion for deciding which papers were worth inspection by our annotators. We tried simple cutoffs, such as at least two rules or at least three rules. See Table 6.2 for details. The results showed that 90% of the CYRENE papers could be captured while discarding 87% of PubMed, or 72% could be captured while discarding 95% of PubMed. While useful, whichever cutoff we used, the resulting set of papers was either too large to handle (often beyond 100,000) or missed far too many of the known relevant papers for the results to be considered complete if applied to PubMed. We experimented with modifications to the rules using a GUI which highlighted the difference in results between different rulesets (see Fig. 6.2).

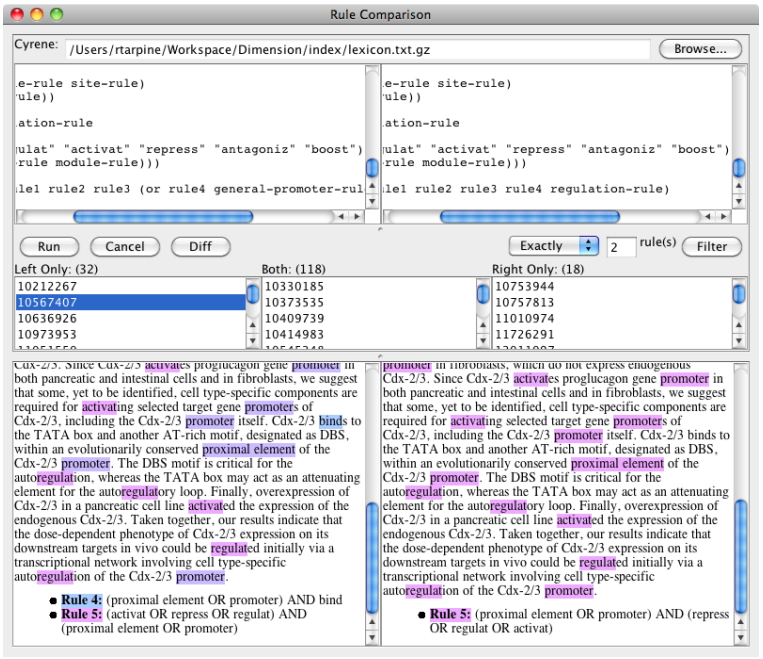


Figure 6.2: GUI for modifying rulesets and visualizing the effects

6.3 Utilizing the full text

Rather than risking missing too many relevant papers, we attempted to use a cutoff resulting in an infeasible number of papers followed by a second stage taking the full-text of the paper into account. The assumption was that the abstract simply doesn't have enough information to discriminate accurately between relevant and irrelevant papers, but the full text must contain the necessary information.

The first difficulty we ran into was simply acquiring the text. While it is well known that PubMed Central (PMC) contains the full text of many articles, its terms of use strictly forbid the use of automated agents to download them:

> You may NOT use any kind of automated process to download articles in bulk from the main PMC site. PMC will block the access of any user who is found to be violating this policy. PMC does have two auxiliary services, the PMC OAI service and the PMC FTP service, that may be used to download certain articles in bulk. The PMC Open Access Subset page explains which articles are available through these services.[2]

Unfortunately, the "certain articles" which are available for bulk download are very few and far between (only 24 out of 506 CYRENE papers (<5%) we tested were available in this manner). After verifying that no such restrictions appeared to be in place for the websites of the journals that most of our CYRENE papers came from, we wrote programs to navigate each site and download the papers given their links from PubMed (the URLs are available programmatically via the NCBI Entrez Programming Utilities [29]). The next step was to extract the text from the documents. We

	0 rules	**≥ 1 rule**	**≥ 2 rules**	**≥ 3 rules**	**All 4 rules**
# PubMed	4,243 (<1%)	20,408 (82.8%)	18,725 (76.0%)	14070 (57.1%)	6763 (27.4%)
# CYRENE	5 (1.1%)	471 (98.9%)	469 (98.5%)	452 (95.0%)	(not recorded)†

† By extrapolating results from a different but similar experiment, I estimate that 80% of the CYRENE papers match all 4 rules (unpublished data)

Table 6.3: Results from applying four modified Davidson Rules to full text

used the PDFTextStripper class from the Apache PDFBox 1.2.1 Java PDF library [32] to extract plain text from the PDFs, and we used the SAXParser class from the CyberNeko HTML Parser 1.9.14 [1] to parse HTML documents and extract their plain text. Approximately 300 PDFs consisted only of scanned images, with no actual text stored inside. These were recognizable due to little or no text being extracted from them. For these, we used Adobe Acrobat Pro 9.4's OCR feature followed by PDFBox text extraction. About 100 of these PDFs could not be processed due to password protection (these documents could be read on screen but not altered), and these were left out of the analysis.

We handled the full text in a similar manner to the title and abstract, applying a modified set of Davidson Rules to them with a higher threshold. Not all the rules were modified. The rules which used a combination of two different terms appearing, such as a form of the word "bind" as well as a form of the word "site", behaved erratically because one term might be found in one section of the document while the other word might occur in a totally different, unrelated section. When applying such rules to abstracts, the problem did not exist because abstracts are so short, forcing all the text to be important and related. At first, we simply removed such rules. At the time we tested this set of rules, we had not yet downloaded all of the papers which passed the title/abstract processing; we had 24,651 PubMed papers and 476 CYRENE papers. The results of this interim evaluation can be seen in Table 6.3. Of the five which weren't matched by any rules, three were PDFs from which we could not extract the text, and two were instances where the file our automated downloader had acquired was not the real paper, because their journals restricted access to them because of their age (i.e., Brown University did not have subscriptions to the journals' legacy content).

There was clearly a beneficial effect, though not as pronounced as the title/abstract method: 95% of the downloaded CYRENE papers could be detected while discarding 43% of the downloaded PubMed papers. This was still not sufficient to meet our needs, however. Being a refinement of the title/abstract results, this doesn't mean that we capture 95% of all CYRENE papers; it implies capturing 95% of the 90% that the first pass accepted. That is only an estimated 86% of the total. And it would still yield over 68,000 papers for our annotators to examine, once all papers were downloaded (57.1% of the 120,000 papers accepted by the title/abstract pass).

We experimented with replacing the AND rules of the title/abstract rules (e.g., "bind" AND "site") with NEAR rules, of the form: "bind" must be within a certain *textual distance* (words or

	0 rules	**≥ 1 rules**	**≥ 2 rules**	**≥ 3 rules**	**≥ 4 rules**	**All 5 rules**
# PubMed	6105 (6.65%)	85643 (93.3%)	85138 (92.8%)	76663 (83.6%)	55899 (60.9%)	33303 (36.3%)
# CYRENE	8 (1.65%)	478 (98.4%)	477 (98.1%)	475 (97.7%)	474 (97.5%)	461 (94.9%)

Table 6.4: Results from applying five NEAR-based Davidson Rules to full text

characters) from the word "site". This requires an ad-hoc definition of the nearness, which we set at 1,400 characters for ease of implementation. We chose 1,400 characters by deciding that the size of an abstract would be a reasonable metric for judging terms to be close. We computed the mean and median sizes of abstracts from our subset of PubMed. We found both (mean: 1388; median: 1588) to be close to 1400 letters, so we decided that terms within 1400 letters of each other are, by our definition, near. The results from using five NEAR-based rules to analyze 91,748 downloaded papers can be see in Table 6.4 (note that each table of results shows the results from analyzing different numbers of papers, so only the percentages should be consulted to compare between them). The results are clearly better than the previous set of Davidson Rules. We captured 95% of the CYRENE papers while discarding 63% of the PubMed papers. Unfortunately this still results in an estimated 43,000 papers for our annotators (36.3% of the 120,000).

It's surprising to see that, for example, more than 60% of the PubMed papers match at least four of the NEAR-based rules. When we examined a few of these papers, we found that (in general) each rule would be matching in different places of the paper. Binding sites might be mentioned in one section, transcription factors might be mentioned in another section, and sometimes even the titles of citations in the bibliography would match rules. We considered attempting to divide each paper into its sections automatically, and (for example) look only in the methodology section, but this proved difficult. Every journal has a different format and would require individual attention, while we search more than one hundred journals. Heuristics which might handle multiple journals without custom code would require recognizing cues in the style, which was not straightforward, especially for PDFs. We had already experienced difficulties in attempting to download papers from each journal which had not yet been totally resolved, and we were not confident that the required work in this new problem would yield sufficient improvement to justify the effort. We decided to focus more on the title/abstract rules, to formalize what they were doing in order to try alternate approaches for defining them.

6.4 Hybrid CLOSE Algorithms

Simple cutoffs implicitly assume that all the rules were equally informative—that any set of three rules, for example, constitutes the same amount of evidence of relevance as any other set of three. This is clearly not true, however. The Obvious Rule (see Fig. 6.1), for example, provides strong evidence of relevance—any paper mentioning a term like "*cis*-regulatory analysis" is likely to be

relevant. While it alone is still not sufficient, it should not take much more information; perhaps also seeing a mention of a reporter construct would be sufficient. The Module Rule, however, is much weaker, especially when it is triggered by vague terms like "enhancer" or "promoter". It requires more much information to recognize that a paper is relevant. To be more precise in judging papers, we would need to enumerate all the combinations that actually imply relevance. This is very difficult to do manually.

We invented a hybrid system that utilizes the lists of related terms compiled by biologists (which we call "concept lists") but combines them in manners determined algorithmically to maximize sensitivity while maintaining specificity. Any system based solely of these concept lists will have an efficient representation: every paper can simply be modeled as the set of concepts it contains. With this simplified view of text, the distinction between relevant and irrelevant papers can be attempted through several models, including standard ones such as support vector machines.

We present an alternative algorithm which we call the Lattice CLOSE Algorithm (LCA). It is motivated by attempting to generate Davidson-esque rules algorithmically, rather than generating a black box based on statistical methods such as SVMs which perform well in practice but do not lead to a better understanding of the problem itself. If we model each paper as a bitstring based on the concepts it contains, then a string such as "101011" represents a document containing four concepts out of a possible six: the first, third, fifth, and sixth concepts (in some arbitrary order). Our Davidson Rules above were monotonic—if a rule would judge a paper relevant due to the presence of certain terms, then adding any additional terms to the document would not change this judgment. Thus certain simple rules can be modeled as bitstrings as well: the minimum set of terms necessary to fulfill the rule. This property of rules also puts a partial ordering on the papers, creating a lattice (any rule matching 101011 also matches 111011, 101111, and 111111; any rule matching 111011 also matches 111111; and any rule matching 101111 also matches 111111).

We can use this lattice view as the data structure for optimization: a set of rules is simply a set of nodes. See Fig. 6.3 for a simple lattice example. This figure illustrates the behavior of rules 1000 and 0110 (highlighted in green): any papers modeled by those two bitstrings as well as eight others (highlighted in magenta) would be matched by those two rules. A benefit of this approach is that the final result can still be interpreted as a set of Davidson-like rules. It can be translated back into English for a biologist to inspect and criticize for improvement.

We first implemented a local search algorithm for generating sets of rules. Each node in the lattice is annotated with two values: the number of CYRENE papers and the number of PubMed papers whose set of concepts is identical to the set represented by the node. The number of CYRENE papers matched by any given rule is then the sum of the CYRENE paper counts of the rule's node in the lattice and its descendants (the number of PubMed papers is computed analogously). We say that a rule is "refined" or made more specific by replacing it with its child nodes. For example, "101011" is refined by replacing it with "111011" and "101111". Using these two rules together matches all of the papers matched by 101011 except those papers who set of concepts is exactly represented by 101011 (i.e., these two rules together match three of the four nodes matched by the original single rule).

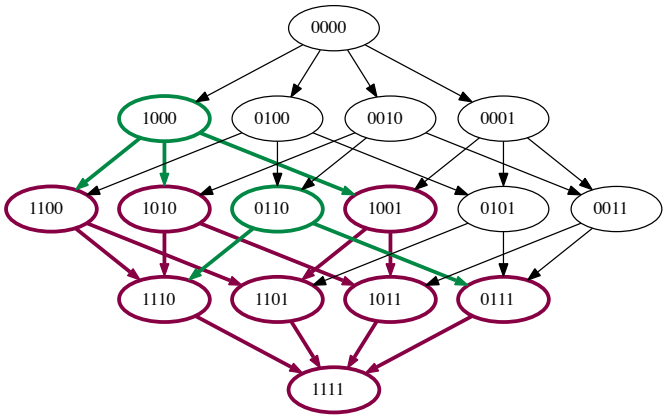


Figure 6.3: Rule lattice constructed from bitstrings of length four (with two rules and their effects highlighted)

The most specific set of rules that matches all of the training papers is the rules implied by the papers themselves. For example, if one training paper contains the concepts 111011 then the rule 111011 is in this set. We start with this set of rules and refine each rule until no improvement can be made (using the F-score defined earlier). At each iteration, we evaluate every rule to determine the increase in the objective score that would be caused by refining it, and we choose the rule with the largest improvement. This is a greedy heuristic, but it performs well in practice.

We experimented with using the concept lists in cooperation with the AdaBoost algorithm [91], but the results were not positive. This may be due to the sparsity of our training data, the small number of possible weak classifiers that can be generated from the concept lists, or due to our specific problem not being amenable to a boosting approach. We believe that the last hypothesis is the correct one, because it is difficult to come up with many weak classifiers for matching abstracts, whether we utilize the concept lists or not. Boosting is a general method for combining the results of many classifiers which individually perform poorly (but better than random) into a single accurate classifier. This requires that many such weak classifiers be available to the boosting algorithm. For handling natural language, individual words which often but not necessarily carry a certain meaning are useful as weak classifiers. In our problem, individual terms are not useful because it is strictly a combination of terms that indicate that a paper fulfills our criteria. Weak classifiers based on individual terms do not perform sufficiently better than random.

Results from using the local search variant of the Lattice CLOSE Algorithm (LCA-LS) have been very promising. Using a set of eleven concept lists (given in Fig. 6.4), 16 rules were selected which match 546 of 567 (96%) training abstracts while matching only 37,920 of 934,606 (4.1%) of the abstracts taken from PubMed. The sixteen rules, using the concept list names given in the previous figure, can be seen in Fig. 6.5. Since we used an F-score as our objective function, we could vary the

α parameter used to determine the tradeoff between precision and recall. The results can be seen in Fig. 6.6.

If a linear scoring function is used instead of an F-score, then an optimal solution can be found by representing the problem as a binary integer programming (BIP) problem. We call this method LCA-BIP. This is due to two properties: (1) a linear objective function can be written as a sum of a sequence of constant scores multiplied by indicator variables for each node in the lattice; and (2) the relationships between nodes in the lattice can be translated into inequalities between variables in the linear program.

A BIP program generated for a two-concept four-node lattice can be seen in Figure 6.7. f is the objective function to be maximized. C_i and P_i are constants for each lattice node i indicating the number of CYRENE papers and PubMed papers associated with the node. α is a nonnegative parameter for determining the weighting between sensitivity and specificity. If $\alpha = 0$ then only sensitivity is taken into account because the P_is effectively vanish from the equation. As α is increased, specificity is taken into account more. The constraints enforce the relationships between nodes in the lattice; for example, $x_{00} \leq x_{01}$ ensures that if node "00" is matched by a rule then node "01" must be as well (if $x_{00} = 1$ then x_{01} must $= 1$ but if $x_{00} = 0$ then x_{01} can be 0 or 1). The output of running a BIP solver will be an assignment to the x_i that agrees with the constraints while maximizing f. To translate the assignment to a set of rules, simply take the set of x_is set to 1 that are minimal in the lattice (i.e, highest, according to the directionality of Figure 6.3). For example, if $x_{00} = 0$ and $x_{01} = x_{10} = x_{11} = 1$, then the minimal x_is are x_{01} and x_{10}, so the optimal rules would be "01" and "10".

While binary integer programming is an NP-complete problem in general, we have found that solutions have always been found in a short amount of time by the MATLAB `bintprog` function for the programs we generate (for example: 15 seconds for a lattice with more than 2,300 nodes representing combinations of thirteen concepts). Even though its objective function was different, we found that this method could result in a higher F-score than the heuristic which attempted to optimize it explicitly. For example, for one dataset LCA-LS found 16 rules which matched 587 of 615 (95%) training abstracts while matching 57,268 of 1,135,969 (5.0%) of the PubMed abstracts, for an F-score of 0.952. LCA-BIP matched 588 training abstracts (95.6%) and 57,384 PubMed abstracts (5.1%), for an F-score of 0.953.

6.5 Evaluation

We evaluated the resulting set by taking a random sample of five hundred papers and examining them to determine how many fit our criteria and contain information for the *cis*-Lexicon. A summary of the results can be seen in Fig. 6.8. Nearly half of the results (43%) were not relevant because they did not analyze transcription factor-encoding genes, the only type we are currently entering into the *cis*-Lexicon. These papers were not examined further to determine whether or not they fit the Davidson Criteria (DC). Similarly, 16% of the papers discussed genes in species other than those

Obvious	Cis-Site	Module
cis-regulatory analys[ie]s	consensus element	enhancer
cis-regulation	cis-acting element	promoter
transcriptional regulation	cis-acting DNA element	regulatory sequence
promoter analys[ie]s	consensus binding site	regulatory module
site-directed mutagenesis	binding site	regulatory region
point mutation	target site	autoregulatory region
mutational analys[ie]s	conserved site	proximal element
specific mutation	sequence motif	distal element
site-directed deletion	consensus site	flanking region
deletion analys[ie]s	binding sequence	flanking sequence
mutation analys[ie]s	response element	flanking DNA
site-specific mutagenesis	sequence element	flanking fragment
deletion study	recognition motif	upstream region
deletion studies	recognition element	bp sequence
deletional analys[ie]s	cis-element	upstream sequence
cis-regulatory inputs	binding factor	
cis-regulatory input	binding protein	
mutational study	regulatory element	
mutational studies	DNA element	
	DNA motif	
	consensus motif	
	functional element	

Tech	Vector	Action
P-factor	GFP	regulat
P-element	CAT	activat
tol2 transposon	lacZ	repress
pronuclear microinjection	beta-galactosidase	antagoni
cell transgenesis	luciferase	boost
viral transgenesis	RFP	synergi
transient transfection	mCherry	
gene transfer	green fluorescent protein	
i-Sce I transposon		
transgenic mouse	**Mutate**	**Binding**
transgenic mice	mutation	bind
co-transfect	mutate	bound
transiently transfect	deletion	
transiently co-transfect	delete	**Characterize**
	mutagenesis	characteriz
		Any-Site
		site
		Reporter
		reporter

Figure 6.4: The eleven concept lists currently used by the Lattice CLOSE Algorithm (LCA)

1. binding AND module AND action
2. vector AND binding AND any-site AND cis-site AND mutate AND action
3. binding AND reporter AND module
4. obvious AND tech AND cis-site AND module
5. any-site AND module AND action
6. vector AND reporter AND tech AND cis-site AND action
7. obvious AND reporter
8. binding AND any-site AND cis-site AND module
9. vector AND binding AND tech AND cis-site AND module AND mutate
10. vector AND tech AND cis-site AND module AND characterize
11. module AND mutate AND action
12. reporter AND cis-site AND mutate AND action
13. binding AND reporter AND any-site AND action
14. vector AND tech AND module AND action
15. binding AND any-site AND module AND mutate AND characterize
16. cis-site AND module AND action

Figure 6.5: Sixteen rules found by LCA-LS

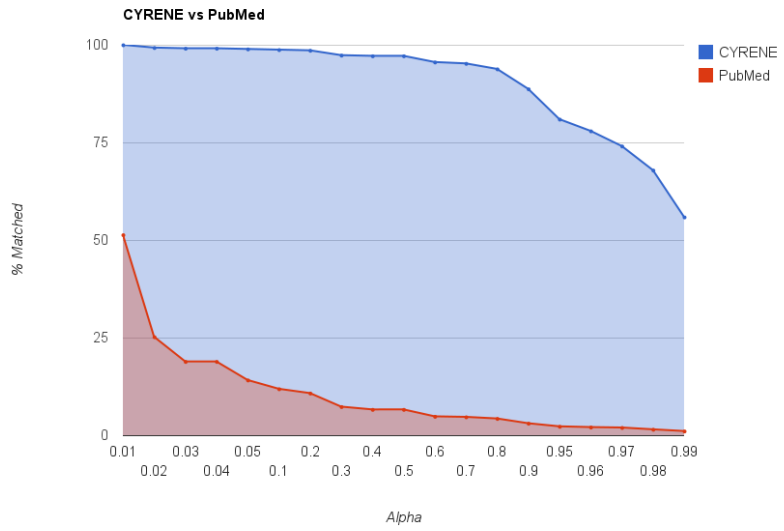


Figure 6.6: Effect of varying the α parameter of the F-score

$$
\begin{aligned}
f(x_{00}, x_{01}, x_{10}, x_{11}) &= (C_{00}x_{00} + C_{01}x_{01} + C_{10}x_{10} + C_{11}x_{11}) \\
&\quad -\alpha(P_{00}x_{00} + P_{01}x_{01} + P_{10}x_{10} + P_{11}x_{11}) \\
&= (C_{00} - \alpha P_{00})x_{00} + (C_{01} - \alpha P_{01})x_{01} \\
&\quad +(C_{10} - \alpha P_{10})x_{10} + (C_{11} - \alpha P_{11})x_{11} \\
x_{00} &\leq x_{01} \\
x_{00} &\leq x_{10} \\
x_{01} &\leq x_{11} \\
x_{10} &\leq x_{11}
\end{aligned}
$$

Figure 6.7: Binary integer program for a four-node CLOSE lattice. See text for explanation.

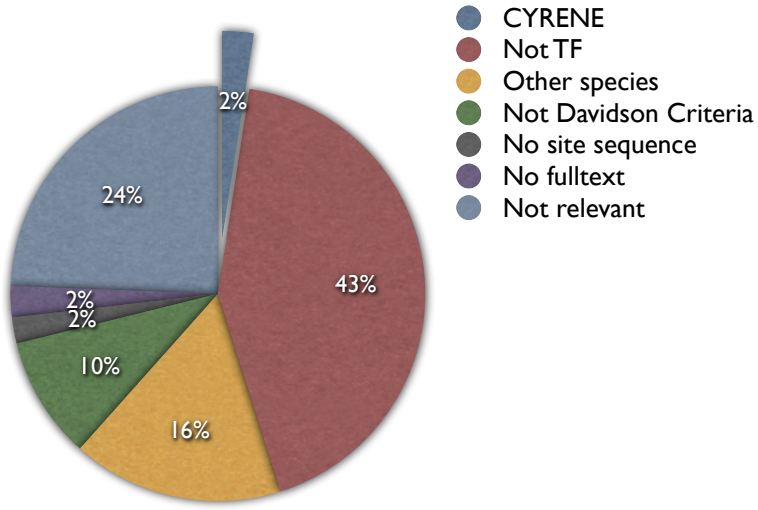


Figure 6.8: Categorization of a random sample of 500 papers selected by LCA-LS

we are focusing on. More significantly, 10% did not fulfill the DC. This is unavoidable to a certain extent, because we know that the abstract is not always sufficient for determining whether the criteria is met. 24% did not contain *cis*-regulatory analysis; these were often papers discussing gene regulation at high level without performing actual experiments. Ultimately, only 2% of the papers contained *cis*-regulatory information fulfilling our experimental criteria. This fraction appears to be an accurate estimation of the entire set of results, because we have found similar percentages for other samples we have analyzed. In recent samples, where we have expanded the set of papers that the algorithm optimizes over to include additional years while still returning the same amount of results ($\approx$ 35,000), the fraction even appears to be closer to 1%.

We have experimented with ways to automatically remove irrelevant papers from the results with an additional pass of what we call "negative rules". The single largest portion of the results we cannot use at this time is the set of papers which discuss non-transcription factor-encoding genes. It is a surprisingly difficult problem to recognize these papers; it is even difficult to recognize gene names. In the Gene Mention Recognition task of the BioCreative (Critical Assessment of Information Extraction systems in Biology) II Workshop, nineteen teams competed to simply find gene names—without even trying to interpret which gene a name refers to—and the highest F_1 score achieved by any team was 0.87 [103]. The authors of the task overview note:

> Technically, finding gene names in text is a kind of named entity recognition (NER) similar to the tasks of finding person names and company names in newspaper text...However, a combination of characteristics, some of which are common to other domains, makes gene names particularly difficult to recognize automatically.[103]

The task of identifying which gene that the text is referring to is an even more difficult problem, known as Gene Normalization [75]. In CLOSE, the problem involves more than recognizing whether

a transcription factor-encoding gene is mentioned in an abstract. Many papers that analyze other types of genes will still mention TFs, because they are the regulators of that non-TF-encoding gene. However, if a non-TF gene is mentioned, especially in the title, the paper is likely to not be relevant. Our annotators noticed a pattern among the non-TF genes that were undergoing *cis*-regulatory analysis. Most of them were enzymes whose name ends in the letters "*ase*" (such as oxygenase, kinase, acetylase, etc). So we devised a quick scan to ignore papers whose title contained a word ending in *ase* other than common dictionary words. This simple heuristic alone removed thousands of papers. We tried other variations, such as words ending in "*oid*" (matching words such as glucocorticoid, retinoid, steroid, etc), but they did not have as much impact. We are in the process of experimenting with similar negative rules for eliminating irrelevant species as well. This is also a tricky problem, because species other than the one under study are often mentioned, to point to information known about related genes in those other species.

Assuming that at about 1% of the results from LCA-LS are indeed relevant, this implies that there are at most $\approx 350 = 35000 \times 0.01$ papers remaining with information relevant. We say "at most" because some genes are analyzed by multiple papers, where the most recent paper therefore contains all of the necessary information, making the previous papers redundant. That estimate agrees well with our initial expectation of about 1000 total relevant papers. Our annotators' rate of judging a paper's relevance has been estimated at 2.5 minutes per paper. At that rate, at 8 hours of work per day, all 35,000 papers can be examined in approximately 6 months by one individual or in a fraction of that time by a team.

Bibliography

[1] Cyberneko html parser. URL `http://nekohtml.sourceforge.net/`.

[2] Are there any restrictions on the use of the material in pmc? can i download a batch of articles from pmc for research or other purposes? URL `http://www.ncbi.nlm.nih.gov/pmc/about/faq/#q2008sep24a`.

[3] D. Aldous. *Probability Approximations via the Poisson Clumping Heuristic*, volume 77 of *Applied Mathematical Sciences*. Springer-Verlag New York, Inc., 1989.

[4] R. B. Altman, C. M. Bergman, J. Blake, C. Blaschke, A. Cohen, F. Gannon, L. Grivell, U. Hahn, W. Hersh, L. Hirschman, L. J. Jensen, M. Krallinger, B. Mons, S. I. O'Donoghue, M. C. Peitsch, D. Rebholz-Schuhmann, H. Shatkay, and A. Valencia. Text mining for biology–the way forward: opinions from leading scientists. *Genome Biol*, 9 Suppl 2:S7, 2008. doi: 10.1186/gb-2008-9-s2-s7.

[5] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. mol. Biol*, 215(3):403–410, 1990.

[6] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389, 1997.

[7] G. Amore and E. H. Davidson. cis-regulatory control of cyclophilin, a member of the ets-dri skeletogenic gene battery in the sea urchin embryo. *Dev Biol*, 293(2):555–64, May 2006. doi: 10.1016/j.ydbio.2006.02.024.

[8] J. P. Balhoff and G. A. Wray. Evolutionary analysis of the well characterized endo16 promoter reveals substantial variation within functional sites. *Proceedings of the National Academy of Sciences of the United States of America*, 102(24):8591–8596, June 2005. ISSN 0027-8424. doi: 10.1073/pnas.0409638102. URL `http://www.ncbi.nlm.nih.gov/pubmed/15937122`. PMID: 15937122.

[9] J. E. Balmer and R. Blomhoff. Evolution of transcription factor binding sites in mammalian gene regulatory regions: handling counterintuitive results. *Journal of Molecular Evolution*,

68(6):654–664, June 2009. ISSN 1432-1432. doi: 10.1007/s00239-009-9238-1. URL http://www.ncbi.nlm.nih.gov/pubmed/19479175. PMID: 19479175.

[10] S. Barolo and J. W. Posakony. Three habits of highly effective signaling pathways: principles of transcriptional control by developmental cell signaling. *Genes Dev*, 16(10):1167–81, May 2002. doi: 10.1101/gad.976502.

[11] P. V. Benos, M. L. Bulyk, and G. D. Stormo. Additivity in protein-dna interactions: how good an approximation is it? *Nucleic Acids Res*, 30(20):4442–51, Oct 2002.

[12] O. G. Berg and P. H. von Hippel. Selection of dna binding sites by regulatory proteins. statistical-mechanical theory and application to operators and promoters. *J Mol Biol*, 193(4): 723–50, Feb 1987.

[13] M. Blanchette and M. Tompa. Discovery of regulatory elements by a computational method for phylogenetic footprinting. *Genome Res*, 12(5):739–48, May 2002. doi: 10.1101/gr.6902.

[14] V. Boeva, J. Clément, M. Régnier, M. A. Roytberg, and V. J. Makeev. Exact p-value calculation for heterotypic clusters of regulatory motifs and its application in computational annotation of cis-regulatory modules. *Algorithms Mol Biol*, 2:13, 2007. doi: 10.1186/1748-7188-2-13.

[15] C. T. Brown, A. G. Rust, P. J. C. Clarke, Z. Pan, M. J. Schilstra, T. De Buysscher, G. Griffin, B. J. Wold, R. A. Cameron, E. H. Davidson, and H. Bolouri. New computational approaches for analysis of cis-regulatory networks. *Dev Biol*, 246(1):86–102, Jun 2002. doi: 10.1006/dbio.2002.0619.

[16] S. Burkhardt and J. Kärkkäinen. Better filtering with gapped q-grams. *Fundamenta informaticae*, 23:1001–1018, 2003.

[17] M. Burrows and D. Wheeler. A block-sorting lossless data compression algorithm. 1994.

[18] R. A. Cameron and E. H. Davidson. Flexibility of transcription factor target site position in conserved cis-regulatory modules. *Developmental Biology*, 336(1):122–135, Dec. 2009. ISSN 1095-564X. doi: 10.1016/j.ydbio.2009.09.018. URL http://www.ncbi.nlm.nih.gov/pubmed/19766623. PMID: 19766623.

[19] R. A. Cameron, S. H. Chow, K. Berney, T. Chiu, Q. Yuan, A. Krämer, A. Helguero, A. Ransick, M. Yun, and E. H. Davidson. An evolutionary constraint: strongly disfavored class of change in DNA sequence during divergence of cis-regulatory modules. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11769–11774, Aug. 2005. ISSN 0027-8424. doi: 10.1073/pnas.0505291102. URL http://www.ncbi.nlm.nih.gov/pubmed/16087870. PMID: 16087870.

[20] J. M. Claverie and S. Audic. The statistical significance of nucleotide position-weight matrix matches. *Comput Appl Biosci*, 12(5):431–9, Oct 1996.

[21] A. J. Courey and S. Jia. Transcriptional repression: the long and the short of it. *Genes Dev*, 15(21):2786–96, Nov 2001. doi: 10.1101/gad.939601.

[22] M. K. Das and H.-K. Dai. A survey of dna motif finding algorithms. *BMC Bioinformatics*, 8 Suppl 7:S21, 2007. doi: 10.1186/1471-2105-8-S7-S21.

[23] M. David, M. Dzamba, D. Lister, L. Ilie, and M. Brudno. Shrimp2: sensitive yet practical short read mapping. *Bioinformatics*, 27(7):1011–2, Apr 2011. doi: 10.1093/bioinformatics/btr046.

[24] E. H. Davidson. *The Regulatory Genome: Gene Regulatory Networks In Development And Evolution*. Academic Press, 1 edition, June 2006. ISBN 9780120885633.

[25] C. G. Elsik, A. J. Mackey, J. T. Reese, N. V. Milshina, D. S. Roos, and G. M. Weinstock. Creating a honey bee consensus gene set. *Genome Biol*, 8(1):R13, 2007. doi: 10.1186/gb-2007-8-1-r13.

[26] G. M. Euskirchen, J. S. Rozowsky, C.-L. Wei, W. H. Lee, Z. D. Zhang, S. Hartman, O. Emanuelsson, V. Stolc, S. Weissman, M. B. Gerstein, Y. Ruan, and M. Snyder. Mapping of transcription factor binding regions in mammalian cells by chip: comparison of array- and sequencing-based technologies. *Genome Res*, 17(6):898–909, Jun 2007. doi: 10.1101/gr.5583007.

[27] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 390–398. IEEE, 2000.

[28] M. Fontaine, S. Burkhardt, and J. Karkkainen. Bdd-based analysis of gapped q-gram filters. *International Journal of Foundations of Computer Science*, 16(6):1121–1134, Dec. 2005.

[29] N. C. for Biotechnology Information. Entrez programming utilities help. URL `http://www.ncbi.nlm.nih.gov/books/NBK25501/`.

[30] N. C. for Biotechnology Information (NCBI). Computation of related citations. URL `http://www.ncbi.nlm.nih.gov/books/NBK3827/#pubmedhelp.Computation_of_Related_Citati`.

[31] C. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial intelligence*, 19(1):17–37, 1982.

[32] T. A. S. Foundation. Apache pdfbox - java pdf library. URL `http://pdfbox.apache.org/`.

[33] M. C. Frith, J. L. Spouge, U. Hansen, and Z. Weng. Statistical significance of clusters of motifs represented by position specific scoring matrices in nucleotide sequences. *Nucleic Acids Res*, 30(14):3214–24, Jul 2002.

[34] M. C. Frith, M. C. Li, and Z. Weng. Cluster-buster: Finding dense clusters of motifs in dna sequences. *Nucleic Acids Res*, 31(13):3666–8, Jul 2003.

[35] J. Fu and W. Lou. *Distribution theory of runs and patterns and its applications: a finite Markov chain imbedding approach*. World Scientific Pub Co Inc, 2003.

[36] S. M. Gallo, D. T. Gerrard, D. Miner, M. Simich, B. D. Soye, C. M. Bergman, and M. S. Halfon. REDfly v3.0: toward a comprehensive database of transcriptional regulatory elements in drosophila. *Nucleic Acids Research*, 39(Database):D118–D123, Oct. 2010. ISSN 0305-1048. doi: 10.1093/nar/gkq999. URL `http://www.nar.oxfordjournals.org/cgi/doi/10.1093/nar/gkq999`.

[37] J. Glaz, V. Pozdnyakov, and S. Wallenstein, editors. *Scan Statistics: Methods and Applications*. Statistics for Industry and Technology. Birkhauser Verlag AG, Boston, MA, 2009.

[38] S. Gray, P. Szymanski, and M. Levine. Short-range repression permits multiple enhancers to function autonomously within a complex promoter. *Genes Dev*, 8(15):1829–38, Aug 1994.

[39] O. L. Griffith, S. B. Montgomery, B. Bernier, B. Chu, K. Kasaian, S. Aerts, S. Mahony, M. C. Sleumer, M. Bilenky, M. Haeussler, M. Griffith, S. M. Gallo, B. Giardine, B. Hooghe, P. V. Loo, E. Blanco, A. Ticoll, S. Lithwick, E. Portales-Casamar, I. J. Donaldson, G. Robertson, C. Wadelius, P. D. Bleser, D. Vlieghe, M. S. Halfon, W. Wasserman, R. Hardison, C. M. Bergman, S. J. Jones, and T. O. R. A. Consortium. ORegAnno: an open-access community-driven resource for regulatory annotation. *Nucleic Acids Research*, 36(Database):D107–D113, Dec. 2007. ISSN 0305-1048. doi: 10.1093/nar/gkm967. URL `http://nar.oxfordjournals.org/content/36/suppl_1/D107.short`.

[40] B. V. Halldórsson, D. Aguiar, R. Tarpine, and S. Istrail. The clark phase-able sample size problem: Long-range phasing and loss of heterozygosity in GWAS. In B. Berger, editor, *Proceedings of the 14th Annual International Conference on Research in Computational Biology (RECOMB10)*, volume 6044 of *Lecture Notes in Computer Science*, pages 158–173. Springer, 2010.

[41] B. V. Halldórsson, D. Aguiar, R. Tarpine, and S. Istrail. The clark phaseable sample size problem: long-range phasing and loss of heterozygosity in GWAS. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, 18(3):323–333, Mar. 2011. ISSN 1557-8666. doi: 10.1089/cmb.2010.0288. URL `http://www.ncbi.nlm.nih.gov/pubmed/21385037`. PMID: 21385037.

[42] E. E. Hare, B. K. Peterson, V. N. Iyer, R. Meier, and M. B. Eisen. Sepsid even-skipped enhancers are functionally conserved in drosophila despite lack of sequence conservation. *PLoS Genetics*, 4(6):e1000106, June 2008. ISSN 1553-7404. doi: 10.1371/journal.pgen.1000106. URL `http://www.ncbi.nlm.nih.gov/pubmed/18584029`. PMID: 18584029.

[43] V. F. Hinman, A. Nguyen, and E. H. Davidson. Caught in the evolutionary act: precise cis-regulatory basis of difference in the organization of gene networks of sea stars and sea urchins.

Dev Biol, 312(2):584–595, Dec 2007. ISSN 1095-564X (Electronic); 0012-1606 (Linking). doi: 10.1016/j.ydbio.2007.09.006.

[44] S. Istrail and E. H. Davidson. Logic functions of the genomic cis-regulatory code. *Proc Natl Acad Sci U S A*, 102(14):4954–4959, April 2005. ISSN 0027-8424. doi: 10.1073/pnas.0409624102. URL `http://dx.doi.org/10.1073/pnas.0409624102`.

[45] S. Istrail, S. B.-T. De-Leon, and E. H. Davidson. The regulatory genome and the computer. *Dev Biol*, 310(2):187–95, Oct 2007. doi: 10.1016/j.ydbio.2007.08.009.

[46] S. Istrail, R. Tarpine, K. Schutter, and D. Aguiar. Practical computational methods for regulatory genomics: A cisGRN-lexicon and cisGRN-browser for gene regulatory networks. In I. Ladunga, editor, *Computational Biology of Transcription Factor Binding*, volume 674 of *Methods in Molecular Biology*, pages 369–399. Springer Science+Business Media, LLC, Humana Press, 2010. doi: 10.1007/978-1-60761-854-6_22. URL `http://www.springerprotocols.com/Abstract/doi/10.1007/978-1-60761-854-6_22`.

[47] C. Jiang, Z. Xuan, F. Zhao, and M. Q. Zhang. TRED: a transcriptional regulatory element database, new entries and other development. *Nucleic Acids Research*, 35(Database issue): D137–D140, Jan. 2007. ISSN 0305-1048. doi: 10.1093/nar/gkl1041. PMID: 17202159 PMCID: 1899102.

[48] D. S. Johnson, W. Li, D. B. Gordon, A. Bhattacharjee, B. Curry, J. Ghosh, L. Brizuela, J. S. Carroll, M. Brown, P. Flicek, C. M. Koch, I. Dunham, M. Bieda, X. Xu, P. J. Farnham, P. Kapranov, D. A. Nix, T. R. Gingeras, X. Zhang, H. Holster, N. Jiang, R. D. Green, J. S. Song, S. A. McCuine, E. Anton, L. Nguyen, N. D. Trinklein, Z. Ye, K. Ching, D. Hawkins, B. Ren, P. C. Scacheri, J. Rozowsky, A. Karpikov, G. Euskirchen, S. Weissman, M. Gerstein, M. Snyder, A. Yang, Z. Moqtaderi, H. Hirsch, H. P. Shulha, Y. Fu, Z. Weng, K. Struhl, R. M. Myers, J. D. Lieb, and X. S. Liu. Systematic evaluation of variability in chip-chip experiments using predefined dna targets. *Genome Res*, 18(3):393–403, Mar 2008. doi: 10.1101/gr.7080508.

[49] P. Jokinen and E. Ukkonen. Two algorithms for approxmate string matching in static texts. *Mathematical Foundations of Computer Science 1991*, pages 240–248, 1991.

[50] Y. Kamachi, M. Uchikawa, A. Tanouchi, R. Sekido, and H. Kondoh. Pax6 and sox2 form a co-dna-binding partner complex that regulates initiation of lens development. *Genes Dev*, 15 (10):1272–86, May 2001. doi: 10.1101/gad.887101.

[51] H. Kirkpatrick, K. Johnson, and A. Laughon. Repression of dpp targets by binding of brinker to mad sites. *J Biol Chem*, 276(21):18216–22, May 2001. doi: 10.1074/jbc.M101365200.

[52] S. Knirr and M. Frasch. Molecular integration of inductive and mesoderm-intrinsic inputs governs even-skipped enhancer activity in a subset of pericardial and dorsal muscle progenitors. *Dev Biol*, 238(1):13–26, Oct 2001. doi: 10.1006/dbio.2001.0397.

[53] F. Lam, R. Tarpine, and S. Istrail. The imperfect ancestral recombination graph reconstruction problem: upper bounds for recombination and homoplasy. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, 17(6):767–781, June 2010. ISSN 1557-8666. doi: 10.1089/cmb.2009.0249. URL `http://www.ncbi.nlm.nih.gov/pubmed/20583925`. PMID: 20583925.

[54] F. Lam, R. Tarpine, and S. Istrail. Conservative extensions of linkage disequilibrium measures from pairwise to multi-loci and algorithms for optimal tagging SNP selection. In V. Bafna and S. C. Sahinalp, editors, *Proceedings of the 15th Annual International Conference on Research in Computational Biology (RECOMB11)*, volume 6577 of *Lecture Notes in Computer Science*, pages 468–482. Springer, 2011.

[55] N. Lam, M. A. Chesney, and J. Kimble. Wnt signaling and ceh-22/tinman/nkx2.5 specify a stem cell niche in c. elegans. *Curr Biol*, 16(3):287–95, Feb 2006. doi: 10.1016/j.cub.2005.12.015.

[56] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biol*, 10(3):R25, Mar 2009. doi: 10.1186/gb-2009-10-3-r25.

[57] D. Latchman. *Eukaryotic transcription factors*. Academic press, 2008.

[58] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton. Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. *Science (New York, N.Y.)*, 262(5131):208–214, Oct. 1993. ISSN 0036-8075. URL `http://www.ncbi.nlm.nih.gov/pubmed/8211139`. PMID: 8211139.

[59] M. Lease and E. Charniak. Parsing biomedical literature. *Natural Language Processing–IJCNLP 2005*, pages 58–69, 2005.

[60] H. Li and R. Durbin. Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, 25(14):1754–60, Jul 2009. doi: 10.1093/bioinformatics/btp324.

[61] H. Li and R. Durbin. Fast and accurate long-read alignment with burrows-wheeler transform. *Bioinformatics*, 26(5):589–95, Mar 2010. doi: 10.1093/bioinformatics/btp698.

[62] L. Li, S. He, J.-M. Sun, and J. R. Davie. Gene regulation by sp1 and sp3. *Biochem Cell Biol*, 82(4):460–71, Aug 2004. doi: 10.1139/o04-045.

[63] L. Li, Q. Zhu, X. He, S. Sinha, and M. S. Halfon. Large-scale analysis of transcriptional cis-regulatory modules reveals both common features and distinct subclasses. *Genome Biol*, 8(6): R101, 2007. doi: 10.1186/gb-2007-8-6-r101.

[64] M. Li, B. Ma, D. Kisman, and J. Tromp. Patternhunter ii: highly sensitive and fast homology search. *Genome Inform*, 14:164–75, 2003.

[65] N. Li and M. Tompa. Analysis of computational approaches for motif discovery. *Algorithms Mol Biol*, 1:8, 2006. doi: 10.1186/1748-7188-1-8.

[66] R. Li, Y. Li, K. Kristiansen, and J. Wang. Soap: short oligonucleotide alignment program. *Bioinformatics*, 24(5):713–4, Mar 2008. doi: 10.1093/bioinformatics/btn025.

[67] R. Li, C. Yu, Y. Li, T.-W. Lam, S.-M. Yiu, K. Kristiansen, and J. Wang. Soap2: an improved ultrafast tool for short read alignment. *Bioinformatics*, 25(15):1966–7, Aug 2009. doi: 10.1093/bioinformatics/btp336.

[68] H. Lin, Z. Zhang, M. Q. Zhang, B. Ma, and M. Li. Zoom! zillions of oligos mapped. *Bioinformatics*, 24(21):2431–7, Nov 2008. doi: 10.1093/bioinformatics/btn416.

[69] C. B. Livi and E. H. Davidson. Regulation of spblimp1/krox1a, an alternatively transcribed isoform expressed in midgut and hindgut of the sea urchin gastrula. *Gene Expr Patterns*, 7 (1-2):1–7, Jan 2007. doi: 10.1016/j.modgep.2006.04.009.

[70] W. J. R. Longabaugh, E. H. Davidson, and H. Bolouri. Computational representation of developmental genetic regulatory networks. *Dev Biol*, 283(1):1–16, Jul 2005. doi: 10.1016/j.ydbio.2005.04.023.

[71] B. Ma, J. Tromp, and M. Li. Patternhunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–5, Mar 2002.

[72] U. Manber and G. Myers. Suffix arrays: a new method for on-line string searches. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 319–327. Society for Industrial and Applied Mathematics, 1990.

[73] C. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2009.

[74] V. Matys, O. V. Kel-Margoulis, E. Fricke, I. Liebich, S. Land, A. Barre-Dirrie, I. Reuter, D. Chekmenev, M. Krull, K. Hornischer, N. Voss, P. Stegmaier, B. Lewicki-Potapov, H. Saxel, A. E. Kel, and E. Wingender. Transfac and its module transcompel: transcriptional gene regulation in eukaryotes. *Nucleic Acids Res*, 34(Database issue):D108–10, Jan 2006. doi: 10.1093/nar/gkj143.

[75] A. A. Morgan, Z. Lu, X. Wang, A. M. Cohen, J. Fluck, P. Ruch, A. Divoli, K. Fundel, R. Leaman, J. Hakenberg, C. Sun, H.-h. Liu, R. Torres, M. Krauthammer, W. W. Lau, H. Liu, C.-N. Hsu, M. Schuemie, K. B. Cohen, and L. Hirschman. Overview of biocreative ii gene normalization. *Genome Biol*, 9 Suppl 2:S3, 2008. doi: 10.1186/gb-2008-9-s2-s3.

[76] T. Nakao and A. Ishizawa. Development of the spinal nerves in the mouse with special reference to innervation of the axial musculature. *Anat Embryol (Berl)*, 189(2):115–38, Feb 1994.

[77] J. Nam, P. Dong, R. Tarpine, S. Istrail, and E. H. Davidson. Functional cis-regulatory genomics for systems biology. *Proceedings of the National Academy of Sciences of the United States of America*, 107(8):3930–3935, Feb. 2010. ISSN 1091-6490. doi: 10.1073/pnas.1000147107. URL `http://www.ncbi.nlm.nih.gov/pubmed/20142491`. PMID: 20142491.

[78] J. Naus. Probabilities for a generalized birthday problem. *Journal of the American Statistical Association*, pages 810–815, 1974.

[79] J. Naus. Approximations for distributions of scan statistics. *Journal of the American Statistical Association*, pages 177–183, 1982.

[80] G. Nong, S. Zhang, and W. Chan. Computing the inverse sort transform in linear time. *ACM Transactions on Algorithms (TALG)*, 7(2):27, 2011.

[81] I. Oda-Ishii, V. Bertrand, I. Matsuo, P. Lemaire, and H. Saiga. Making very similar embryos with divergent genomes: conservation of regulatory mechanisms of otx between the ascidians halocynthia roretzi and ciona intestinalis. *Development*, 132(7):1663–74, Apr 2005. doi: 10.1242/dev.01707.

[82] G. Ostlund, T. Schmitt, K. Forslund, T. Köstler, D. N. Messina, S. Roopra, O. Frings, and E. L. L. Sonnhammer. Inparanoid 7: new algorithms and tools for eukaryotic orthology analysis. *Nucleic Acids Res*, 38(Database issue):D196–203, Jan 2010. doi: 10.1093/nar/gkp931.

[83] P. A. Pevzner and S. H. Sze. Combinatorial approaches to finding subtle signals in dna sequences. *Proc Int Conf Intell Syst Mol Biol*, 8:269–78, 2000.

[84] E. Portales-Casamar, S. Thongjuea, A. T. Kwon, D. Arenillas, X. Zhao, E. Valen, D. Yusuf, B. Lenhard, W. W. Wasserman, and A. Sandelin. Jaspar 2010: the greatly expanded open-access database of transcription factor binding profiles. *Nucleic Acids Res*, 38(Database issue):D105–10, Jan 2010. doi: 10.1093/nar/gkp950.

[85] A. Ransick and E. H. Davidson. cis-regulatory processing of notch signaling input to the sea urchin glial cells missing gene during mesoderm specification. *Dev Biol*, 297(2):587–602, Sep 2006. doi: 10.1016/j.ydbio.2006.05.037.

[86] Y. Regev, M. Finkelstein-Landau, R. Feldman, M. Gorodetsky, X. Zheng, S. Levy, R. Charlab, C. Lawrence, R. A. Lippert, Q. Zhang, and H. Shatkay. Rule-based extraction of experimental evidence in the biomedical domain: the kdd cup 2002 (task 1). *SIGKDD Explor. Newsl.*, 4:90–92, December 2002. ISSN 1931-0145. doi: http://doi.acm.org/10.1145/772862.772874. URL `http://doi.acm.org/10.1145/772862.772874`.

[87] J. Rister and C. Desplan. Deciphering the genome's regulatory code: the many languages of DNA. *BioEssays: News and Reviews in Molecular, Cellular and Developmental Biology*, 32 (5):381–384, May 2010. ISSN 1521-1878. doi: 10.1002/bies.200900197. URL `http://www.ncbi.nlm.nih.gov/pubmed/20394065`. PMID: 20394065.

[88] S. M. Rumble, P. Lacroute, A. V. Dalca, M. Fiume, A. Sidow, and M. Brudno. Shrimp: accurate mapping of short color-space reads. *PLoS Comput Biol*, 5(5):e1000386, May 2009. doi: 10.1371/journal.pcbi.1000386.

[89] M. P. Samanta, W. Tongprasit, S. Istrail, R. A. Cameron, Q. Tu, E. H. Davidson, and V. Stolc. The transcriptome of the sea urchin embryo. *Science*, 314(5801):960–2, Nov 2006. doi: 10.1126/science.1131898.

[90] E. W. Sayers, T. Barrett, D. A. Benson, E. Bolton, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. Dicuccio, S. Federhen, M. Feolo, I. M. Fingerman, L. Y. Geer, W. Helmberg, Y. Kapustin, S. Krasnov, D. Landsman, D. J. Lipman, Z. Lu, T. L. Madden, T. Madej, D. R. Maglott, A. Marchler-Bauer, V. Miller, I. Karsch-Mizrachi, J. Ostell, A. Panchenko, L. Phan, K. D. Pruitt, G. D. Schuler, E. Sequeira, S. T. Sherry, M. Shumway, K. Sirotkin, D. Slotta, A. Souvorov, G. Starchenko, T. A. Tatusova, L. Wagner, Y. Wang, W. J. Wilbur, E. Yaschenko, and J. Ye. Database resources of the national center for biotechnology information. *Nucleic Acids Res*, 40(Database issue):D13–25, Jan 2012. doi: 10.1093/nar/gkr1184.

[91] R. Schapire. The boosting approach to machine learning: An overview. *LECTURE NOTES IN STATISTICS-NEW YORK-SPRINGER VERLAG-*, pages 149–172, 2003.

[92] M. Schindler. A fast block-sorting algorithm for lossless data compression. In *Proceedings of the Conference on Data Compression*, volume 469. Citeseer, 1997.

[93] T. D. Schneider and R. M. Stephens. Sequence logos: a new way to display consensus sequences. *Nucleic Acids Res*, 18(20):6097–100, Oct 1990.

[94] P. T. Shannon, D. J. Reiss, R. Bonneau, and N. S. Baliga. The gaggle: an open-source software system for integrating bioinformatics software and data sources. *BMC Bioinformatics*, 7:176, 2006. doi: 10.1186/1471-2105-7-176.

[95] H. Shatkay. Hairpins in bookstacks: information retrieval from biomedical text. *Brief Bioinform*, 6(3):222–38, Sep 2005.

[96] H. Shatkay and R. Feldman. Mining the biomedical literature in the genomic era: an overview. *J Comput Biol*, 10(6):821–55, 2003. doi: 10.1089/106652703322756104.

[97] H. Shatkay, N. Chen, and D. Blostein. Integrating image data into biomedical text categorization. *Bioinformatics*, 22(14):e446–53, Jul 2006. doi: 10.1093/bioinformatics/btl235.

[98] H. Shatkay, F. Pan, A. Rzhetsky, and W. J. Wilbur. Multi-dimensional classification of biomedical text: toward automated, practical provision of high-utility text to diverse users. *Bioinformatics*, 24(18):2086–93, Sep 2008. doi: 10.1093/bioinformatics/btn381.

[99] K. Shimizu and K. Tsuda. Slidesort: all pairs similarity search for short reads. *Bioinformatics*, 27(4):464–70, Feb 2011. doi: 10.1093/bioinformatics/btq677.

[100] A. D. Smith, Z. Xuan, and M. Q. Zhang. Using quality scores and longer reads improves accuracy of solexa read mapping. *BMC Bioinformatics*, 9:128, 2008. doi: 10.1186/1471-2105-9-128.

[101] A. D. Smith, W.-Y. Chung, E. Hodges, J. Kendall, G. Hannon, J. Hicks, Z. Xuan, and M. Q. Zhang. Updates to the rmap short-read mapping software. *Bioinformatics*, 25(21):2841–2, Nov 2009. doi: 10.1093/bioinformatics/btp533.

[102] J. Smith and E. H. Davidson. A new method, using cis-regulatory control, for blocking embryonic gene expression. *Dev Biol*, 318(2):360–5, Jun 2008. doi: 10.1016/j.ydbio.2008.02.056.

[103] L. Smith, L. K. Tanabe, R. J. n. Ando, C.-J. Kuo, I.-F. Chung, C.-N. Hsu, Y.-S. Lin, R. Klinger, C. M. Friedrich, K. Ganchev, M. Torii, H. Liu, B. Haddow, C. A. Struble, R. J. Povinelli, A. Vlachos, W. A. Baumgartner, Jr, L. Hunter, B. Carpenter, R. T.-H. Tsai, H.-J. Dai, F. Liu, Y. Chen, C. Sun, S. Katrenko, P. Adriaans, C. Blaschke, R. Torres, M. Neves, P. Nakov, A. Divoli, M. Maña-López, J. Mata, and W. J. Wilbur. Overview of biocreative ii gene mention recognition. *Genome Biol*, 9 Suppl 2:S2, 2008. doi: 10.1186/gb-2008-9-s2-s2.

[104] G. D. Stormo. Dna binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23, Jan 2000.

[105] R. Tarpine and S. Istrail. On the concept of cis-regulatory information: From sequence motifs to logic functions. In A. Condon, D. Harel, J. N. Kok, A. Salomaa, and E. Winfree, editors, *Algorithmic Bioprocesses*, Natural Computing Series, pages 731–742. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-88869-7. doi: 10.1007/978-3-540-88869-7_36. URL `http://dx.doi.org/10.1007/978-3-540-88869-7_36`.

[106] M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. De Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent, V. J. Makeev, A. A. Mironov, W. S. Noble, G. Pavesi, G. Pesole, M. Régnier, N. Simonis, S. Sinha, G. Thijs, J. van Helden, M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, and Z. Zhu. Assessing computational tools for the discovery of transcription factor binding sites. *Nat Biotechnol*, 23(1):137–44, Jan 2005. doi: 10.1038/nbt1053.

[107] C. Trapnell, L. Pachter, and S. L. Salzberg. Tophat: discovering splice junctions with rna-seq. *Bioinformatics*, 25(9):1105–11, May 2009. doi: 10.1093/bioinformatics/btp120.

[108] R. Turner. Software projects. URL `http://www.cs.umbc.edu/~turner/index-projects.html`.

[109] K. Wang, D. Singh, Z. Zeng, S. J. Coleman, Y. Huang, G. L. Savich, X. He, P. Mieczkowski, S. A. Grimm, C. M. Perou, J. N. MacLeod, D. Y. Chiang, J. F. Prins, and J. Liu. Mapsplice: accurate mapping of rna-seq reads for splice junction discovery. *Nucleic Acids Res*, 38(18): e178, Oct 2010. doi: 10.1093/nar/gkq622.

[110] W. W. Wasserman and A. Sandelin. Applied bioinformatics for the identification of regulatory elements. *Nat Rev Genet*, 5(4):276–287, Apr. 2004. ISSN 1471-0056. doi: 10.1038/nrg1315. URL `http://dx.doi.org/10.1038/nrg1315`.

[111] D. Weese, A.-K. Emde, T. Rausch, A. Döring, and K. Reinert. Razers–fast read mapping with sensitivity control. *Genome Res*, 19(9):1646–54, Sep 2009. doi: 10.1101/gr.088823.108.

[112] A. G. West, M. Gaszner, and G. Felsenfeld. Insulators: many functions, many mechanisms. *Genes Dev*, 16(3):271–88, Feb 2002. doi: 10.1101/gad.954702.

[113] W. J. Wilbur, A. Rzhetsky, and H. Shatkay. New directions in biomedical text annotation: definitions, guidelines and corpus construction. *BMC Bioinformatics*, 7:356, 2006. doi: 10.1186/1471-2105-7-356.

[114] C. Yuh, C. T. Brown, C. B. Livi, L. Rowen, P. J. C. Clarke, and E. H. Davidson. Patchy interspecific sequence similarities efficiently identify positive cis-regulatory elements in the sea urchin. *Developmental Biology*, 246(1):148–61, June 2002. ISSN 0012-1606. doi: 12027440. URL `http://www.ncbi.nlm.nih.gov/pubmed/12027440`. PMID: 12027440.

[115] C. Yuh, E. R. Dorman, M. L. Howard, and E. H. Davidson. An otx cis-regulatory module: a key node in the sea urchin endomesoderm gene regulatory network. *Developmental Biology*, 269(2):536–51, May 2004. ISSN 0012-1606. doi: 15110718. URL `http://www.ncbi.nlm.nih.gov/pubmed/15110718`. PMID: 15110718.

[116] C. H. Yuh, H. Bolouri, and E. H. Davidson. Genomic cis-regulatory logic: experimental and computational analysis of a sea urchin gene. *Science*, 279(5358):1896–902, Mar 1998.

[117] C. H. Yuh, H. Bolouri, and E. H. Davidson. Cis-regulatory logic in the endo16 gene: switching from a specification to a differentiation mode of control. *Development*, 128(5):617–29, Mar 2001.

[118] J. Zeitlinger, R. P. Zinzen, A. Stark, M. Kellis, H. Zhang, R. A. Young, and M. Levine. Whole-genome chip-chip analysis of dorsal, twist, and snail suggests integration of diverse patterning processes in the drosophila embryo. *Genes Dev*, 21(4):385–90, Feb 2007. doi: 10.1101/gad.1509607.

[119] R. W. Zeller, J. D. Griffith, J. G. Moore, C. V. Kirchhamer, R. J. Britten, and E. H. Davidson. A multimerizing transcription factor of sea urchin embryos capable of looping dna. *Proc Natl Acad Sci U S A*, 92(7):2989–93, Mar 1995.

[120] R. P. Zinzen, C. Girardot, J. Gagneur, M. Braun, and E. E. M. Furlong. Combinatorial binding predicts spatio-temporal cis-regulatory activity. *Nature*, 462(7269):65–70, Nov. 2009. ISSN 1476-4687. doi: 10.1038/nature08531. URL `http://www.ncbi.nlm.nih.gov/pubmed/19890324`. PMID: 19890324.