Sequence comparison using a relational database approach

Andre Bergholz
Institut fur Informatik
Humboldt-Universitat zu Berlin
Unter den Linden 6
D - 10099 Berlin, Germany
bergholz@dbis.informatik.hu-berlin.de

Stephan Heymann
Max-Delbruck-Centrum fur Molekulare Medizin
Robert-Rossle-Strase 10
D - 13122 Berlin, Germany
heymann@mdc-berlin.de

Jorg A. Schenk
Max-Delbruck-Centrum fur Molekulare Medizin
Robert-Rossle-Strase 10
D - 13122 Berlin, Germany
jschenk@mdc-berlin.de

Johann Christoph Freytag
Institut fur Informatik
Humboldt-Universitat zu Berlin
Unter den Linden 6
D - 10099 Berlin, Germany
freytag@dbis.informatik.hu-berlin.de

Abstract

A variety of heterogenous data sources is available in the field of molecular biology. Our focus lies on the biological sequence data, i. e. data maintained in collections like EMBL or SWISS-PROT. We propose a relational model based on an Entity-Relationship approach for this discourse world. This is the foundation of a flexible architecture useful for a variety of purposes, e. g. for sequence comparison. In this paper we present our first application system within this architecture, the sequence analysis tool NNSAT. The description of this system is accompanied by a short review of the data model and a section on the problem of sequence comparison in which we propose a small modification of the Needleman and Wunsch alignment method.

Keywords: Gene and protein sequences, ER model, Relational model, Sequence alignment, Applications on relational systems

1 Introduction

The field of molecular biology offers a variety of challenges for computer scientists. Among them are finding suitable representations for the various data, visualization of data or the derivation of new knowledge based on existing data. From the biological point of view our main research interest is the extraction of knowledge about the functionality of proteins given only the primary structure (sequence) of the protein or the DNA. In this paper we describe a sequence comparison tool built within a flexible architecture that allows us not only to search for homologous sequences in the usual way but also to test own hypotheses. The paper is organized as follows. Section 2 provides an overview of our motivation and related work in the area. In Section 3 we briefly summarize our relational database approach for biological sequence data. In Section 4 we present a theoretical result about sequence comparison and choose an alignment algorithm for our purposes. Section 5 is the main part of this paper describing the analysis tool we built. Conclusions and future work are given in Section 6.

2 Motivation and related work

The main data we are interested in consist of gene and protein sequences. There are many projects dealing with this kind of data but unfortunately none of them is suitable for our purposes. These projects can be categorized as follows:

1. data collection projects: In this group there are the data collections that list the massive amount of sequence data material available. Usually, these data collections consist of a set of ASCII files in a well defined format. Typical data collections include EMBL ([20]) and GenBank ([7]) for DNA and RNA sequences, and SWISS-PROT ([5]) and PIR ([6]) for protein sequences.

2. sequence comparision programs: Many algorithms for homology sequence comparison have been developed and implemented. Among the most well-known are the method of Needleman and Wunsch ([18]), the FASTA ([16],[19]) and the BLAST program ([4]).

3. sequence analysis packages: Here we have complete program packages that combine the above two groups such that common analysis methods can be applied to common data material. PC/Gene ([17]) and HUSAR ([22]) are the ones mostly used.

4. ‘‘huge’’ projects: World-wide initiatives and cooperation has led to huge genome projects. The data material stored within these projects far exceeds the core biological data. Leading projects include the Integrated Genome Database (IGD, [21]) based on the ACeDB database ([23]) and the Genome Database (GDB, [12]).

The group related closest to our approach is the third one. However, those program packages do not provide a database interface. Rather, the various subprograms output their specific results in files. This inevitably leads to proprietory file management as there will be lots of result files from various searches, lots of fetched sequence files etc. Furthermore, one cannot simply add one’s own application programs since there is no way to effectively access the data material.

With our work we fill this gap. For this reason we built up a relational database storing biological sequence data from common data collections. This step results in a flexible interface to the data, thus providing a solid foundation for developing application programs.

3 Biological Sequences Integrated: A relational database approach

Data collection projects generate huge amounts of DNA and protein data every day. The amount of data made available grows exponentielly. However, most of this data is still organized in ASCII files. We chose two of the largest collections, EMBL ([20]) for gene data and SWISS-PROT ([5]) for protein data, as sources for our project. (Recall that one aim of the database was to integrate both gene and protein sequence data.) Currently, the EMBL collection consists of about 1.1 million entries and SWISS-PROT of about 60000 entries.

Our Entity-Relationship approach to biological sequence data has been discussed previously in [8]. Figure 1 summarizes our entities leaving out attribute specification.

Based on this model we implemented a database called Biological Sequences Integrated (BSI) using the relational DBMS Sybase ([1]). The database was filled via the Embedded SQL interface of the Sybase system. The EMBL and SWISS-PROT ASCII files were scanned using awk ([3]) scripts. In case we want to add data from other collections we must only add a new awk script scanning the new source. The current size of the BSI database is about two gigabytes.

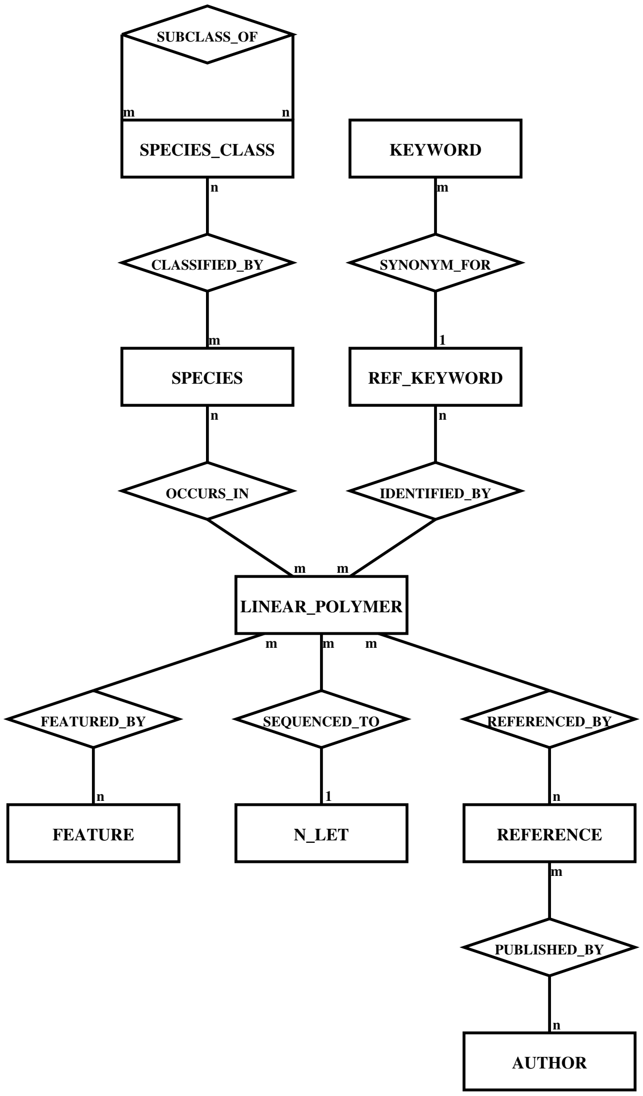

Figure 1. An ER-approach (complete, without attributes)

By creating this database we now have a standard interface that allows us to fast extract the bits of information we are interested in.

4 The problem of sequence comparison

There exists a large number of sequence comparison tools. Ordinary string matching algorithms like the one introduced by Knuth, Morris and Pratt ([15]) could be used to find homologous sequence segments. However, we are faced with several problems complicating the matter. First, it is state of the art not only to compare characters but to associate property vectors to the characters representing amino acids. Such property vectors have been proposed by Bork and Grunwald ([9], [10]). Second, introducing gaps in sequences is useful because during millions of years insertions or deletions of nucleotides in gene sequences could have occured so that now related gene segments are not necessarily of the same length. Introducing gaps dramatically increases the complexity. As will be proved elsewhere, there are

$$\sum_{i=0}^{\min(l_1,l_2)} \binom{\max(l_1,l_2)+i}{i} \binom{\max(l_1,l_2)}{\min(l_1,l_2)-i}$$

possible alignments of two sequences with the lengths l_1 and l_2, respectively. The basic idea behind the prove is that an alignment A of a fixed length l is uniquely identified by the positions of the gaps in the first sequence and the positions where two sequence elements are aligned. Compared to the $\max(l_1,l_2) - \min(l_1,l_2) + 1$ alignments that exist without gaps this is quite a large number. If, for reason of simplicity, we assume $l_1 = l_2$ and call this length l then the number of possible alignments is between 2^l and $(2l+2)^l$.

Now, of course, no program can compute all alignments possible in reasonable time. Programs like FASTA or BLAST perform local sequence comparisons. The algorithm by Needleman and Wunsch performs global comparision, but skips some alignments. Restrictions are that at least one of the first sequence elements and one of the last must be aligned to sequence elements. Furthermore, an alignment with a sequence element of the first sequence being aligned to a gap in the second at one position and a gap in the first sequence being aligned to a sequence element of the second sequence at the position right next cannot be expressed by the Needleman and Wunsch matrix path.

Nevertheless, we decided to use the Needleman and Wunsch method in our analysis tool. The complexity of this method, as described in the original work ([18]), is $O(l^3)$ with l denoting the length of the sequences (for reason of simplicity it is assumed that both sequences have the same length). A $O(l^2)$ matrix has to be filled and each matrix element takes $O(l)$ time for the comparison operations. However, if one keeps track of the past comparisons in the rows and columns, the complexity of the computation of one matrix element can be reduced to $O(1)$ and thus the whole complexity to $O(l^2)$ ([13]).

Since we also want to use the search algorithm in the field of protein-protein interaction we extended the Needleman and Wunsch method to be able to cope with neighbourhood. Recall that the Needeman and Wunsch method consists of two steps. At first a matrix with the two sequences as axes is initialized. $M(i,j)$ then represents how well the i-th element of the first sequence fits to the j-th element of the second. In the second step, an optimal path through the matrix from the bottom right to the top left is computed. This path represents which sequence elements are aligned and where the gaps occur. Our neighbourhood consideration leads to a modification of the first step. $M(i,j)$ now also represents weighted information on how well the i-th element of the first sequence fits to the $j-1$-th, $j+1$-th, $j-2$-th etc. element of the second as well as how good the j-th element of the second sequence fits to the $i-1$-th, $i+1$-th etc. element of the first. The weighting is done by the factor $1/(d+1)$ where d is the distance of the respective sequence elements.

5 An integrated search tool

Our first application that uses the newly created database BSI is a search tool with the following functionality currently availabe:

- integration of the relational database BSI
- support of various file formats (currently plain sequence, EMBL and SWISS-PROT) and manual input
- generation of random sequences (with individual probabilities for the various sequence elements)
- elementary functions for DNA and protein, e. g. translation or anti-sense strand of a given DNA
- two variants of the Needleman and Wunsch algorithm with complexity $O(l_1 l_2)$ (one considers the neighbourhood of elements)
- support of various alignment matrices (currently identity, standard DNA, PAM250, Property-identity, Anticodon, Mirror and Invers-complementarity)

This software is implemented using an object-oriented approach. We extract the relevant bits of our relational model and transform the entities into classes. The attributes are transformed to instance variables. We still keep the distiction between the concept *NLet* and the concept *LinearPolymer*. In our relational database approach, the entity LINEAR_POLYMER was introduced in order to represent a database entry in its complete complexity, while an N_LET only represents the actual sequence itself. For *LinearPolymer* we keep the attributes `source` (was DATABASE_ID in the relational model), `localID`, `primaryAccession` and `description`. Furthermore, an instance of *LinearPolymer* also contains an instance of the class *NLet* as member. Instance variables of *NLet* include `sequence`, `length` and `composition` (the composition was not part of the relational model, but is derived from the sequence instead). In addition, the class *DNANLet* contains two more instance variables, namely an ambiguity flag indicating whether the sequence is only made of the ''true'' nucleotides `a`, `c`, `g` and `t` and the information density. We defined the information density in [14] as

$$H = -\frac{1}{l} log_4 \frac{1^{\Sigma_{i=\mathtt{a,c,g,t}} n_i} \cdot 2^{\Sigma_{i=\mathtt{m,r,w,s,y,k}} n_i} \cdot 3^{\Sigma_{i=\mathtt{v,h,d,b}} n_i} \cdot 4^{n_\mathtt{n}}}{4^l}$$

where l is the length of the sequence and the $n_\mathtt{i}$ denote the frequency of occurence for the nucleotide i. This information density gives an indication for the grade of degeneration of a DNA sequence. (The degenerated DNA alphabet contains characters denoting alternatives for a position in a DNA, e. g. `m` stands for `a` or `c`.) A summary of the biological bits we represent in our analysis tool is shown in figure 2.

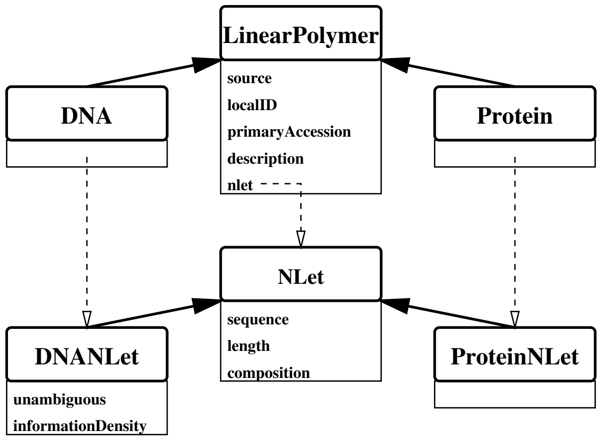

Figure 2. Object-oriented modeling of biological sequence information

In addition, we defined a class *Alignment* containing a two-dimensional array for the actual alignment as well as instance variables for the length of the alignment, the number of aligned n-lets and the quality of the alignment (represented for each position and as sum over the whole alignment).

Apart from the biologically relevant structures there are other classes that are mostly needed for program organization or additional analysis. We use concepts like *LinkedList*, *Array* or *Matrix* that are useful for a variety of purposes. For instance, our *Array* class comes with a statistics part. Figure 3 gives an overview of all introduced classes.

The application software NNSAT (No Name Sequence Analysis Tools) runs in textmode. This guarantees easy portablity since our final platform is still uncertain as we are working at various locations. For this interface two classes, *TextInterface* and *TextExecute*, are introduced. The

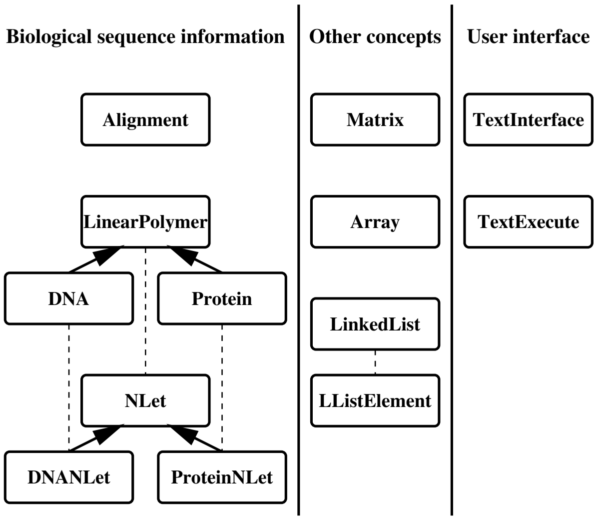

Figure 3. Object-oriented approach to sequece analysis application

class *TextInterface* contains an instance variable `action` keeping the from the user demanded action. The class *TextExecute* provides the actions. So the user communicates only to an instance of the class *TextInterface*. Via a set of predefined signals the user choice is directed to an instance of the class *TextExecute* so that the chosen subprogram can be executed. Figure 4 summarizes the idea.

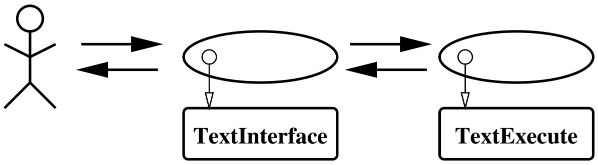

Figure 4. The user interface

An advantage of this approach is that if at some time in the future we decide to give the program a graphical interface the class *TextExecute* can still be used for a transitional period. Subprograms can then be executed in seperate windows.

As implementation language we chose Objective-C ([11]). Like C++, Objective-C is an object-oriented extension of the C language. Objective-C was developed in the mid-80s and became known as the basic language for the NeXTStep ([2]) operating system. Classes and message passing are provided in a Smalltalk-like manner. Objects are dynamically typed. The complete type information including instance variables and accessible methods become known only at run-time. This flexibility provides e. g. an easier implementation of the ideas in figure 2 where the `nlet` instance variable of class *DNA* should not be of type

NLet but rather of type *DNANLet*. Furthermore, a linked list can easily contain objects of various types. Finally, another nice feature of Objective-C is the category concept. A category is a bag of methods that can be added to a class. Thus, one can avoid that the class hierarchy gets endless due to improving search algorithms since now no ''useless'' derivations of the class *NLet* have to be made.

The communication to the database BSI is realized via an independent database interface. It provides operations such as selection, insertion and update. An obvious advantage is that porting to another database system only requires a reimplementation of the interface instead of reimplementing the complete application program. This approach also allows us to run the program without a database connection for many useful subroutines of NNSAT do not require it. Figure 5 shows the basic architecture.

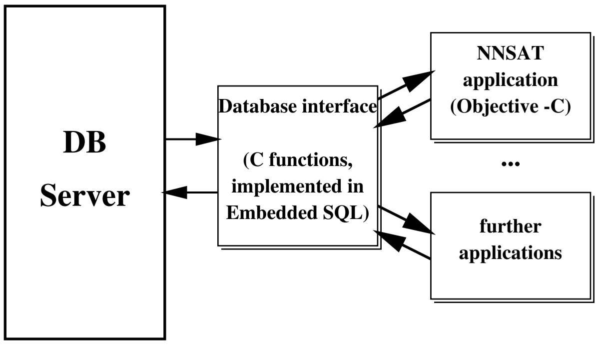

Figure 5. Communication to the database

We have used our system for homology searches as well as for testing of hypotheses within the field of protein-protein interaction. With the latter we had notable success on problems arising in virology and oncology which will be published elsewhere.

6 Conclusion and future work

The field of molecular biology produces huge amount of data material. This data material is very difficult to handle, because of its heterogenity, its only partly known structure, synonym and homonym problems, its redundancy and its inconsistency. This gives computer scientists the task to look for more flexible ways to deal with data. In this project, we focus on a smaller discourse world, namely that of biological sequence data. We provided a relational database interface to the data collections EMBL and SWISS-PROT.

On top of the relational database interface we designed an application software in an object-oriented manner that allows us to execute basic oprations on gene and protein sequences as well as to search for related sequences using various alignment matrices. This software called NNSAT is implemented in Objective-C, an object-oriented extension of the C language providing high flexibility. The application communicates with our relational database BSI via an independent interface using Embedded SQL. We have already used the software intensively to test hypotheses on protein-protein interaction.

Our system resembles closely well-known tools like HUSAR. The drawback is that our system currently has much less functionality then these comparable systems but the advatage is that all users have a direct access to all pieces of data. The database system can handle all kinds of queries and new applications can easily be added.

In the future, the functionality of the system will gradually be extended. Our current focus, however, lies on extending our software with a WWW-interface. We plan to provide a database interface as well as the possibility to perform alignment searches.

Acknowledgement

The authors wish to thank Michael Rybczak and Heinz Werner for their extensive technical support.

References

[1] Sybase, Inc., `http://www.sybase.com`.
[2] NeXT Software, Inc., `http://www.next.com`.
[3] A. V. Aho, B. W. Kernighan, and P. J. Weinberger. *The awk Programming Language*. Addison-Wesley, 1988.
[4] S. F. Altschul, W. Gish, W. Miller, E. M. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403--410, 1990.
[5] A. Bairoch and B. Boeckmann. The SWISS-PROT protein sequence data bank, recent developments. *Nucleic Acids Research*, 21(13):3093--3096, 1993.
[6] W. C. Barker, D. G. George, H.-W. Mewes, F. Pfeiffer, and A. Tsugita. The PIR-international databases. *Nucleic Acids Research*, 21(13):3089--3092, 1993.
[7] D. Benson, D. J. Lipman, and J. Ostell. GenBank. *Nucleic Acids Research*, 21(13):2963--2965, 1993.
[8] A. Bergholz, J. A. Schenk, S. Heymann, and J. C. Freytag. An entity-relationship approach to biological sequence data. *Symposium on Complex Genetic Diseases*, 1995.
[9] P. Bork and C. Grunwald. A method for property pattern searches in protein sequence databases, demonstrated by detection of gtp-binding sites. *Stud. Biophys.*, 129:231--240, 1989.
[10] P. Bork and C. Grunwald. Recognotion of different nucleotide-binding sites in primary structures using a property-pattern approach. *Eur. J. Biochem.*, 191:347--358, 1990.
[11] B. J. Cox and A. J. Novobilski. *Object-Oriented Programming: An Evolutionary Approach*. Addison-Wesley, 2nd edition, 1991.

[12] K. H. Fasman, S. I. Letovsky, R. W. Cottingham, and D. T. Kingsbury. Improvements to the GDB^{TM} human genome data base. *Nucleic Acids Research*, 24(1):57--63, 1996.

[13] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705--708, 1982.

[14] S. Heymann, J. A. Schenk, A. Bergholz, and B. Micheel. What else is in a genome - minimum information, a new concept. *in preparation*, 1997.

[15] D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323--350, 1977.

[16] D. J. Lipman and W. R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227:1435--1441, 1985.

[17] J. Moore, A. Engelberg, and A. Bairoch. Using PC/Gene for protein and nucleic acid analysis. *Biotechniques*, 6(6):566--572, June 1988.

[18] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443--453, 1970.

[19] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparision. *Proc. Natl. Acad. Sci.*, 85:2444--2448, 1988.

[20] C. M. Rice, R. Fuchs, D. G. Higgins, P. J. Stoehr, and G. N. Cameron. The EMBL data library. *Nucleic Acids Research*, 21(13):2967--2971, 1993.

[21] O. Ritter. The integrated genomic database. *Computational Methods in Genome Research*, pages 57--73, 1994.

[22] M. Senger, K. H. Glatting, O. Ritter, and S. Suhai. X-HUSAR, an x-based graphical interface for the analysis of genomic sequences. *Comput. Methods Programs Biomed.*, 46(2):131--141, 1995.

[23] J. Thierry-Mieg and R. Durbin. Syntactic definitions for the ACEDB data base. *Technical Report MRC-LMB xx.92*, 1992.