Bioinformatics Adventures in Database Research

Jinyan Li, See-Kiong Ng, and Limsoon Wong

Laboratories for Information Technology
21 Heng Mui Keng Terrace, Singapore 119613
Email: {jinyan, skng, limsoon}@lit.a-star.edu.sg

Abstract. Informatics has helped launch molecular biology into the genomic era. It appears certain that informatics will remain a major contributor to molecular biology in the post-genome era. We discuss here data integration and datamining in bioinformatics, as well as the role that database theory played in these topics. We also describe LIMS as a third key topic in bioinformatics where advances in database system and theory can be very relevant.

1 The Great Challenges in Bioinformatics

Bioinformatics is key to genome projects. The main challenges are to improve the content and utility of databases, to develop better tools for data generation, capture, and annotation, to develop and improve tools and databases for functional studies, to develop and improve tools for representing and analysing sequence similarity and variation, and to create mechanisms to support effective approaches for producing robust software that can be widely shared. In the 1990s, a lot of databases were produced by the numerous mapping and sequencing initiatives. The hot topics were problems of managing and integrating these databases, and of comparing and assembling sequences contained in these databases. In the 2000s, several genomes have been completely sequenced and we enter the era of post-genome knowledge discovery. The hot topics are problems of recognizing specific features and functions of DNA and protein sequences, understanding their role in diseases, and their interaction with each other and their environment.

We focus on two stories. The first story concerns data integration challenges in bioinformatics in the 1990s. We narrate the triumph of the theory of query languages and the Kleisli system [6, 39, 9] in addressing these challenges. The second story concerns knowledge discovery challenges in bioinformatics in the 2000s. We narrate the tantalizing success of emerging patterns and the PCL technique [23, 22] in addressing some of these challenges. This story is still an ongoing saga.

The paper is organized as follows. Section 2 presents the story of Kleisli. It begins with a discussion (Subsection 2.1) on what are the problems in the integrating biological data, and names (Subsection 2.2) the key features of a successful general solution to this problem. The key features are nested relational data model (Subsection 2.3), self-describing data exchange format (Subsection 2.4), thin wrapper (Subsection 2.5), and high-level query language (Subsection 2.6).

Section 3 presents the story of PCL. It begins with a discussion (Subsection 3.1) on knowledge discovery from biomedical data, and uses the recognition of translation initiation site from mRNA sequences as an example (Subsection 3.2). Then several general techniques for feature generation (Subsection 3.3) and feature selection (Subsection 3.4) are introduced. After that, PCL is introduced and applied to the recognition of translation initiation sites (Subsection 3.5). The algorithmic aspects underlying PCL are considered in Subsection 3.6.

Section 4 wraps up the paper with a discussion on laboratory information management systems or LIMS. We describe the key considerations for LIMS in the management of data and processes in a genome laboratory. While we have not conducted extensive research on this topic, we feel that LIMS is an area where advances in database system and theory, especially those pertaining to workflows, can offer significant benefit to molecular biology research.

2 From Theory of Query Languages to Biological Data Integration

2.1 Integration: What are the problems?

Many kinds of data are used in research in biology and medicine. They include laboratory records, literatures, nucleic acid and amino acid sequences, microarray output, and so on, as well as many computational tools for analysis of these data. We use the term data sources to refer to both databases and computational analysis tools. These data sources were mostly designed and built by biologists for their own individual use. This root leads to a significant legacy problem [2]. At the same time, it is realised that these data sources can also used in combination to answer questions that cannot be answered by any single data source [10]. This is thus a great challenge for the research and development of a general data integration system that can handle the heterogeneous, complex, and geographically dispersed nature of biomedical data sources. Let us introduce an "impossible query" from the US Department of Energy 1993 Bioinformatics Summit Report [10] that made this challenge famous.

Example 1. The query was to find, as many as possible, non-human homologs of each gene on a given cytogenetic band of a given human chromosome. Basically, this query means that for each gene in a particular position in the human genome, find DNA sequences from non-human organisms that are similar to it.

source	type	location	remarks
GDB	Sybase	Baltimore	Flat tables, SQL joins, cytogenetic band info
Entrez	ASN.1	Bethesda	Nested tables, keyword retrieveal, homolog info

The challenge in this query lies not in the biology, but in the data sources as summarised in the table above. At the time it was posed, the main database containing cytogenetic band information was the GDB [29], which was a Sybase system. GDB did not store the DNA sequences that were needed for determining homologs. The sequences needed were kept in GenBank, which at that time could only be accessed via the ASN.1 version of Entrez [33]. Entrez also kept precomputed homologs. So this query needed the integration of GDB (a relational database located in Baltimore) and Entrez (a non-relational "database" located in Bethesda). This query was considered "impossible" as there was at that time no system that could work across the bioinformatics sources involved due to their heterogeneity, complexity, and geographical locations. □

2.2 Integration: Solution

A large number of solutions have been proposed to the data integration challenge in biomedicine. The main features of the most general and successful amongst these solutions are: data models that support nesting of structures, simple self-describing data exchange formats, thin wrappers, high-level query languages, good query optimizers, host language interfaces, and backend stores that support nesting of structures. The notions of data models, query languages, query optimizers, and backend stores are classic ideas in relational database management systems. Even the host language interface idea has manifested itself in the development of relational database systems as embedded SQL, JDBC and ODBC for quite some time. The arena of biomedical data integration requires that these ideas be extended to nested relations or other equivalent forms. These ideas still assume that the database system is the center of the world and that all data are kept within the database system. On the other hand, the ideas of data exchange formats and thin wrappers have appeared explicitly only in the mid 90s [28]. Furthermore, these ideas assume that data can be external to the database system and must be communicated and accessed in different ways.

It is by embracing these generalized old ideas of the database world and newer ideas outside of the database world that Kleisli [9] became the first successful general solution to the data integration problem in biomedicine. In the next subsections, we describe Kleisli and the influence of query language theory on its design and implementation.

2.3 Nested relational data model

The nested relational data model was first proposed by Makinouchi [26] and enjoyed a fruitful evolution in a sequence of works [18, 35] in the database world. However, the inspiration for the nested relational data model as in the Kleisli system arises from a different consideration [5]. In modern programming language theory, the key concepts are orthogonality and universality. These concepts together mean that for every data type, there should be constructs to build data of that type and constructs to take apart data of that type so that certain equations hold, and furthermore, so long as typing constraints are respected, these constructs can be freely combined. When the constructs for building sets and records are freely combined, nested sets arise naturally in Kleisli.

A data model is focused on the logical structure of data. The lexical aspect of data is irrelevant. The application semantic aspect of data is also irrelevant. A data model also comes with a query language for manipulating that logical structure. The link between the logical structure and application semantics of the data is captured by queries that are written a query language. The link between the logical structure and the lexical structure of the data is captured by a data exchange format. We discuss these links next.

2.4 Self-describing data exchange format

The concept of a self-describing data exchange format is implicit in programming languages that support type inference [8]. In these programming languages, every linguistic construct has an unambiguous meaning and type. The linguistic constructs for building data thus gives a self-describing data exchange format. The same concept arises explicitly, but much later, in database research [28].

The aspects of a self-describing data exchange format are: “lexical”—how to parse? “logical”—what is the structure? and “semantics”—what is the meaning? The lexical and logical layers are the important aspects of an exchange format. The better they are designed, the easier it is to exchange information. The semantics layer is not needed to accomplish the exchange of data between two systems. In addition, the lexical layer should support multiple logical layers. That is, objects of different structures can be laid out in a data stream using the same lexical layer. “Printing” and “parsing” are the two operations that map between the lexical and logical layers. “Transmit” is the operation to move data laid out in the data exchange format between two systems. All good data exchange systems provide the equivalent of “print”, “transmit”, and “parse”.

Example 2. We first present a poor data exchange format. Here is a GenPept report of a F-box protein. The GenPept report is an inconvenient data exchange format as its logical and lexical layers are not explicit. It is also not a versatile data exchange format as it cannot be used for any other kind of data.

```
LOCUS       T41727        577 aa        PLN       03-DEC-1999
DEFINITION  F-box domain protein Pof3p - fission yeast
ACCESSION   T41727
PID         g7490551
VERSION     T41727  GI:7490551
DBSOURCE    pir: locus T41727;
            summary: #length 577 #weight 66238 ...
KEYWORDS    .
SOURCE      fission yeast.
  ORGANISM  Schizosaccharomyces pombe
            Eukaryota; Fungi; Ascomycota; ...
REFERENCE   1  (residues 1 to 577)
  AUTHORS   Lyne,M., Wood,V., Rajandream,M.A., ...
  TITLE     Direct Submission
  JOURNAL   Submitted (??-JUN-1998) to the EMBL Data Library
FEATURES             Location/Qualifiers
     source          1..577
                     /organism="Schizosaccharomyces pombe"
                     /db_xref="taxon:4896"
     Protein         1..577
                     /product="F-box domain protein Pof3p"
ORIGIN
   1 mnnyqvkaik ektqqylskr kfedaltfit ktieqepnpt ...
```
□

The table below is a better exchange format used in Kleisli [40]. The logical and lexical layers are separated so that a generic “parser” can be used for any data laid out in this exchange format. The logical layer conforms to the nested relational model. This allows a generic parser to be built into Kleisli so that it can directly parse and manipulate any data conforming to this exchange format. Also, Kleisli outputs in this format by default.

logical layer	lexical layer	remarks
Booleans	true, false	
Numbers	123, 123.123	Positive numbers
	–123, –123.123	Negative numbers
Strings	”a string”	String is inside double quotes
Records	(#l_1: O_1, ..., #l_n: O_n)	Record is inside round brackets
Sets	{ O_1, ..., O_n }	Set is inside curly brackets
Lists	[O_1, ..., O_n]	List is inside square brackets

Example 3. The F-box report from Example 2 looks like this in the data exchange format used by Kleisli:

```
(#uid: 7490551,
 #title: "F-box domain protein Pof3p - fission yeast",
 #accession: "T41727",
 #common: "fission yeast.",
 #organism: (#genus: "Schizosaccharomyces",
        #species: "pombe",
        #lineage: ["Eukaryota", "Fungi", "Ascomycota", ...]),
 #feature: {(#name: "source", #start: 0, #end: 576,
        #anno: [(#anno_name: "organism",
                 #descr: "Schizosaccharomyces pombe"),
                (#anno_name:"db_xref", #descr:"taxon:4896")]),
       (#name: "Protein", #start: 0, #end: 576,
        #anno: [(#anno_name: "product",
                 #descr: "F-box domain protein Pof3p")])},
 #sequence: "MNNYQVKAIKEKTQQYLSKRKFEDALTFITKTIEQEPNPTID...")
```

This format change is simple and has a few positive consequences: (a) the boundaries of different nested structure becomes explicit, and (b) the lexical and logical aspects are no longer mixed up. As a result, a parser specific to the GenPept format is not needed. □

2.5 Thin Wrappers

A wrapper is an interface that (a) conveys requests from one system to a second system, (b) conveys replies from the second system to the first system, and (c) performs any necessary lexical/logical mapping. Whether a wrapper is easy to implement or not is affected by the following factors: the impedance between the data models of the two systems, the complexity of the access protocols of the two systems, and the suitability of the programming language chosen for implementing the wrapper.

As most of the data sources are flat files, the complexity of the access protocol for them is low. In systems such as Kleisli that are based on a nested relational data model, the data model impedance is also low. In systems such as Kleisli that has a simple data exchange format—which also conforms to the nested relational data model—the complexity of the protocol required for conveying replies to them is low. These two advantages for wrapper development in the Kleisli system can be attributed to its strong query language theory foundation [39]. Its nested relational data model is motivated by orthogonality considerations in query language design and its exchange format follows from type inference considerations in query language design.

In contrast, writing wrappers for a system like IBM's DiscoveryLink [16] requires more effort. DiscoveryLink relies on the flat relational data model, which causes high impedance mismatch with most of the biological data sources as these sources generally have nested structures. DiscoveryLink also lacks a simple data exchange format, which causes high complexity to get replies into the system. DiscoveryLink also insists on wrappers being written in C or C++, which are not convenient for parsers.

2.6 High-level query language

Kleisli supports two high-level query languages: CPL [39] and sSQL [6]. Both are nested relational query languages and have the same expressive power. The former was the original query language of the Kleisli system and was inspired by research on the connection between comprehension and monads [37, 4]. The latter was added to the Kleisli system more recently by popular demand and was based on SQL. Let us show the advantage of using a high level language like sSQL with the following query in sSQL: `select title from DATA`. All of the following low-level details that a Perl programmer has to handle explicitly—if the same query is implemented in Perl—are handled automatically by the Kleisli compiler for sSQL: input, lexical-logical mapping, type checking, termination, optimization, output, etc. These low-level details can be handled automatically because the right compromises have been made in sSQL that reduce its complexity and yet without hurting its ability to express queries important for large-scale data manipulations.

The core of CPL and sSQL has their roots in $\mathcal{NRC}$, a nested relational calculus. The syntax of $\mathcal{NRC}$ is summarised below and its semantics is given in earlier papers [5, 38].

$$\frac{}{f_t^s : s \to t} \qquad \frac{}{x^t : t} \qquad \frac{e_1 : t \quad e_2 : t}{e_1 = e_2 : \mathcal{B}}$$

$$\frac{}{\textit{true} : \mathcal{B}} \qquad \frac{}{\textit{false} : \mathcal{B}} \qquad \frac{e : \mathcal{B} \quad e_1 : t \quad e_2 : t}{\textit{if } e \textit{ then } e_1 \textit{ else } e_2 : t} \qquad \frac{e_1 : t_1 \quad e_2 : t_2}{(e_1, e_2) : t_1 \times t_2} \qquad \frac{e : t_1 \times t_2}{\pi_1\ e : t_1} \qquad \frac{e : t_1 \times t_2}{\pi_2\ e : t_2}$$

$$\frac{e : t}{\{e\} : \{t\}} \qquad \frac{e_1 : \{t\} \quad e_2 : \{t\}}{e_1 \cup e_2 : \{t\}} \qquad \frac{}{\{\}^t : \{t\}} \qquad \frac{e_1 : \{t_1\} \quad e_2 : \{t_2\}}{\bigcup\{e_1 \mid x^{t_2} \in e_2\} : \{t_1\}}$$

The design of $\mathcal{NRC}$ is such that for every data type that it supports, it provides constructs for building data of that type and constructs for taking apart data of that type in a way that satisfies universality and orthogonality conditions. For the record or pair type: (e_1, e_2) forms a pair; and $\pi_1\ e$ and $\pi_2\ e$ extracts the first and second component of the pair e. For the set type: $\{\}$, $\{e\}$, and $e_1 \cup e_2$ form the empty set, the singleton set, and the union of two sets respectively; and $\bigcup\{e_1 \mid x \in e_2\}$ is the big union of $g(o_1)$, ..., $g(o_n)$ if $g(x) = e_1$ and $\{o_1, ..., o_n\} = e_2$. Also, f stands for any additional functions that $\mathcal{NRC}$ can be extended with later. We denote extension to $\mathcal{NRC}$ in brackets. For example, $\mathcal{NRC}(\mathcal{Q}, +, \cdot, -, \div, \sum, \leq^{\mathcal{Q}}, =)$ means $\mathcal{NRC}$ extended with rational numbers, arithmetics, summation, and the linear order on rational numbers.

Theorem 1 (Cf. [5, 38, 24]).

1. $\mathcal{NRC} = \textit{Thomas\&Fischer} = \textit{Schek\&Scholl} = \textit{Colby}$.

2. *$\mathcal{NRC} = FO(=) = RelationalAlgebra$ over flat relations.*
3. *$\mathcal{NRC}(\mathcal{Q}, +, \cdot, -, \div, \sum, \leq^{\mathcal{Q}}, =)$ over flat relations is "entry-level" SQL.* □

These results say that $\mathcal{NRC}$ and its various extensions or restrictions, as the case may be, are equivalent to various classical query languages. They also bring out three important points. Firstly, $\mathcal{NRC}$ has enough expressive power for large-scale data manipulations commonly expected in database systems. Secondly, $\mathcal{NRC}$ can be straightforwardly extended by any number of functions $f : s \to t$ of any input-output type as shown in its syntax, whereas the classical query languages above are lacking in orthogonality in their design and cannot accomodate such extensions so easily. This ease of extension is a crucial when handling the integration of biomedical data sources that often include a lot of specialized access and analysis functions. Lastly, the proofs of these results all rely on a rewrite system based on the equation of monads. This rewrite system is also used as the starting point of Kleisli's query optimizer [39].

While $\mathcal{NRC}$ is amenable to automated analysis and optimization, it is not as amendable for a human to read. For example, the cartesian product of two sets is expressed in $\mathcal{NRC}$ as $\bigcup\{\bigcup\{\{(x, y)\} \mid y \in S\} \mid x \in R\}$. This is clumsy. Hence, we need an alternative surface syntax to be for users of the Kleisli system. A popular choice is sSQL, which is based on SQL and is very human readable. E.g., the same cartesian product is expressed as: `select (x,y) from R x, S y`. Queries in sSQL are converted into $\mathcal{NRC}$ via the Wadler equalities [37] linking monads and comprehensions. We present now the sSQL solution to Example 1.

Example 4. `sybase-add (#name:"GDB", ...);`
```
  create view L from locus_cyto_location using GDB;
  create view E from object_genbank_eref using GDB;
  select #accn: g.#genbank_ref, #nonhuman-homologs: H
  from L c, E g,
    {select  u
     from g.#genbank_ref.na-get-homolog-summary u
     where not(u.#title string-islike "%Human%"),
           not(u.#title string-islike "%H.sapien%")} H
  where c.#chrom_num = "22" andalso g.#object_id = c.#locus_id
   andalso not (H = { });
```
The first three lines connect to GDB and map two tables in GDB to Kleisli. These two tables could then be referenced within Kleisli as if they were two locally defined sets, `locus` and `eref`. The next few lines extract from these tables the accession numbers of genes on Chromosome 22, use the Entrez function `aa-get-homolog-summary` to get their homologs, and filter these homologs for non-human ones. □

We see that the integration of GDB and Entrez is smooth, and the handling of input, output, and lexical-logical mapping are fully taken care of. It is also very efficient. The Kleisli optimizer is able to migrate the join, selection, and projections on the two GDB tables into a single efficient access to GDB. The accesses to Entrez are also automatically made concurrent. These are just some advantages that a well-implemented system with a high-level query language can bring over a poor man's Perl library.

3 From Emerging Patterns to Biological Datamining

3.1 Prediction: What are the problems?

Microarrays are now being used to measure the expression level of thousands of genes simultaneously [14]. We can expect reasonably that gene expression data measured by microarrays or other means will soon be part of a patient's medical records. The analysis of medical records is aimed mainly at diagnosis, prognosis, and treatment planning. Here we are looking for patterns that are (a) valid: they are also observed in new data with high certainty; (b) novel: they are not obvious to experts and provide new insights; (c) useful: they enable reliable predictions; and (d) understandable: they pose no obstacle in their interpretation. In short, a classification technique that is both highly accurate and highly understandable is more desirable.

Gene expression profiles and medical records represent the spectrum of biomedical data that possess explicit signals or features—such as expression level of individual genes—where traditional machine learning and datamining techniques appear to be directly useable. However, these techniques have performance limitation in terms of the number of signals or features that they can handle in their input. A modern microarray such as the Affymetrix U95A GeneChip can produce a gene expression profile consisting over the expression levels of over 12000 genes. Such a large number of features is beyond what can be comfortably handled by most of these techniques. So techniques for dimension reduction or feature selection are also needed.

Not all biomedical data contain explicit signals or features. DNA sequences, protein sequences, and so on represent the spectrum of biomedical data that possess no explicit features. E.g., a genomic sequence is typically just a string consisting of the letters "A", "C", "G", and "T" in an apparently random order. And yet a genomic sequence possesses biologically meaningful structure points such as transcription start site, translation initiation site, and so on that are associated with the primary structure of genes. Recognition of these biological structure points in a genomic sequence is an important bioinformatics application. Traditional machine learning and datamining techniques cannot be directly apply to this type of recognition problems. So there is a need to adapt these existing techniques to this type of problems and a need for good techniques for generating explicit features underlying such structure points.

In the next subsections, we use the recognition of translation initiation site to illustrate the challenges above—feature generation, selection, and integration for prediction—of knowledge discovery in biological data.

3.2 Recognition of Translation Initiation Sites

Proteins are synthesized from mRNAs by a process called translation. The region at which the process initiates is called the Translation Initiation Site (TIS). The TIS recognition problem is to correctly identify the TIS in an mRNA or cDNA sequence. In Zien et al. [43] and Hatzigeorgiou [17], accurate TIS prediction is obtained by complex methods: A careful engineering of kernel functions for a support vector machine (SVM) and a multi-step integrated artificial neural network (ANN). Recently, one of us (Wong) shows that good performance can be obtained by simple feature generation and selection followed by a variety of standard machine learning methods [42]. In what follows, we repeat this approach [42], but we use features generated from a translation of the mRNAs into the corresponding amino acids instead of the mRNAs directly, and we use PCL as the classification algorithm instead of traditional machine learning methods such as artificial neural network [32], Bayesian classifier [21], decision tree induction [31], support vector machine [36].

We use the vertebrate dataset provided by Pedersen and Nielsen [30]. This dataset was also used by Zien et al. [43]. These sequences are processed by removing possible introns and joining the exons [30] to obtain the corresponding mRNA sequences. From these sequences, only those with an annotated TIS, and with at least 100 upstream nucleotides as well as 100 downstream nucleotides are selected. The sequences are filtered to remove those belonging to same gene families, homologous genes from different organisms and sequences added multiple times to the database. Since the dataset is processed DNA, the TIS site is ATG. In total there are 13375 ATG sites. Of these possible ATG start sites, 3312 (24.76%) are the true start TIS, while the other 10063 (75.23%) are non-TIS. At each of these ATG sites, we extract a sequence segment consisting of (a) 100 nucleotides up-stream or upto the first occurrence of an ATG up-stream, whichever is shorter; and (b) 100 nucleotides down-stream. These 13375 sequence segments are the inputs upon which we perform the recognition of TIS.

3.3 Feature Generation

The sequences are not suitable for direct application of machine learning techniques, as these techniques rely on explicit signals of high quality. It is necessary to devise various sensors for such signals, so that given a sequence segment, a score is produced to indicate the possible presence of such a signal in that sequence segment. A general technique for producing these sensors is the idea of k-grams frequency. A k-gram is simply a pattern of k consecutive letters, which could be amino acid symbols or nucleic acid symbols. K-grams can also be restricted those in-frame ones. Each k-gram and its frequency in the said sequence fragment becomes a signal. Another general technique for producing these sensors is the idea of position-specific k-gram. The sensor simply reports what k-gram is seen in a particular position in the sequence fragment.

Example 5. For TIS prediction, we use a combination of the general techniques mentioned above. First, we divide each of our 13375 sequence segments into an up-stream part, i.e., to the left of the ATG, and a down-stream part, i.e., to the right of the ATG. Then we consider 3-grams that are in-frame, i.e., those 3-grams aligned to the ATG. Those in-frame 3-grams that code for amino acids are converted into amino acid letters. Those in-frame 3-grams that are stop codons are converted into a special letter symbolizing a stop codon. Then we generate the follow numeric features: `UP-X` (resp. `DOWN-X`), which counts the number of times the amino acid letter `X` appears in the up-stream part (resp. down-stream), for `X` ranging over the standard 20 amino acid letters. We also generate the following Boolean features: `UP-ATG`, which indicates an in-frame ATG occurs in the up-stream part; `DOWN-STOP` (resp. `UP-STOP`), which indicates a stop codon occurs in the down-stream part (resp. up-stream); and UPn-N (resp. DOWNn-N), which indicates the nucleotide `N` occurs in position n up-stream (resp. down-stream), for N ranging over the 4 standard nucleotide letters (A, C, G, T) and 6 generalizations (AorC, AorG, AorT, CorG, CorT, CorG), and for n ranging over the 10 up-stream (resp. down-stream) positions closest to the ATG. Thus for each of our 13375 sequence segments, we have $(20 \times 2) + 1 + 2 + (10 \times 10 \times 2) = 243$ features. □

3.4 Feature Selection

As can be seen, the techniques described above can generate a large number of features for each sequence segment. Clearly, not every feature is important. Using only the more important features has the advantage of avoiding noise and speeding up subsequent construction of the recognition model. It is thus often desirable to discard weaker features. A number of general techniques can be used for this purpose [25, 13].

Here, let us use the entropy method [13]. The basic idea of this method is to filter out those features whose value distributions are relatively random. For the remaining features, this method can automatically find some cut points in these features' value ranges such that the resulting value intervals of every feature can be maximally distinguished. If each value interval induced by the cut points of a feature contains only the same class of samples, then this partitioning by the cut points of this feature has an entropy value of zero. The smaller a feature's entropy is, the more discriminatory it is. Formally, let T partition the set S of examples into the subsets S_1 and S_2. Let there be k classes $C_1, \cdots, C_k$. Let $P(C_i, S_j)$ be the proportion of examples in S_j that have class C_i. The *class entropy* of a subset $S_j, j = 1, 2$ is defined as:

$$Ent(S_j) = -\sum_{i=1}^{k} P(C_i, S_j) log(P(C_i, S_j)).$$

Suppose the subsets S_1 and S_2 are induced by partitioning a feature A at point T. Then, the *class information entropy* of the partition, denoted $E(A, T; S)$, is given by:

$$E(A, T; S) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2).$$

A binary discretization for A is determined by selecting the cut point T_A for which $E(A, T; S)$ is minimal amongst all the candidate cut point. The same process can be applied recursively to S_1 and S_2. The *Minimal Description Length Principle* is used to stop partitioning. That is, recursive partitioning within a set of values S stops iff

$$Gain(A, T; S) < \frac{log_2(N-1)}{N} + \frac{\delta(A, T; S)}{N},$$

where N is the number of values in the set S, $Gain(A, T; S) = Ent(S) - E(A, T; S)$, $\delta(A, T; S) = log_2(3^k - 2) - [k \cdot Ent(S) - k_1 \cdot Ent(S_1) - k_2 \cdot Ent(S_2)]$, and k_i is the number of class labels represented in the set S_i. We choose those features with lowest entropy values.

Example 6. Applying the entropy method to the features generated in Example 5 for the whole data set, the 10 features of lowest entropy are UP-ATG, DOWN-STOP, DOWN-A, `UP3-AorG`, `DOWN-V`, `UP-A`, `DOWN-E`, `DOWN-L`, `DOWN-D`, and `UP-G`. We also perform three-fold cross validation. The data set is divided into three groups of the same size, where the proportion of true TIS to non-TIS is kept the same as the original undivided data set. In each fold, one of the groups is held out as test cases and the other two groups are used as training cases. In the training of each fold, we apply the entropy method to each group of training cases to select the 10 features of lowest entropy for that fold. It turns out that, for each fold, the same 10 features listed above are picked. Some of these 10 features also make good biological sense. The `UP-ATG` feature makes sense because it is uncommon for an in-frame up-stream ATG to be near a translation initiation site, as this runs counter to the scanning model of eukaryotic protein translation [19]. The `DOWN-STOP` feature makes sense because it is uncommon for an in-frame stop codon to be near a translation initiation site, as this implies an improbably short protein product. The `UP3-AorG` feature makes sense because it is consistent with the well-known Kozak consensus signature observed at translation initiation sites [19].

$\square$

3.5 Prediction: Solution

In the field of machine learning, there are many good prediction methods including k-nearest neighbours (k-NN) [7], C4.5 [31], SVM [36], NB [21], etc. C4.5 induces from training data rules that are easy to comprehend. However, it may not have good performance if real decision boundary underlying the data is not linear. NB uses Bayesian rules to compute a probabilistic summary for each class. A key assumption used in NB is that the underlying features are statistically independent. However, this is not appropriate for gene expression data analysis as genes involved in an expression profile are often closely related and appear not to be independent. k-NN assigns a testing sample the class of its nearest training sample in terms of some non-linear distance functions. Even though k-NN is intuitive and has good performance, it is not helpful for understanding complex cases in depth. SVM uses non-linear kernel functions to construct a complicated mapping between samples and their class labels. SVM has good performance, but it functions as a black box. Similarly, traditional datamining methods that look for high frequency patterns are not useful on these data. E.g., if you use these methods in the Singapore General Hospital, they will produce totally useless patterns such as "everyone here has black hair and black eyes."

We are therefore motivated to seek a classification method that enjoys the twin advantages of high accuracy and high comprehensibility. We have been developing a novel classification method called PCL [23, 22] for Prediction by Collective Likelihood of emerging patterns. This method focuses on (a) fast techniques for identifying patterns whose frequencies in two classes differ by a large ratio [11], which are the so-called emerging patterns; and on (b) combining these patterns to make decision. For PCL, we use only emerging patterns that are most general and have infinite ratio.

Basically, the PCL classifier has two phases. Given two training datasets D^P (instances of class P) and D^N (instances of class N) and a test sample T, PCL first discovers two groups of most general emerging patterns from D^P and D^N. Denote the most general emerging patterns of D^P as, EP_1^P, EP_2^P, $\cdots$, EP_i^P, in descending order of frequency. Denote the most general emerging patterns of D^N as EP_1^N, EP_2^N, $\cdots$,

EP_j^N, in descending order of frequency. Suppose the test sample T contains these most general emerging patterns of D^P: $EP_{i_1}^P$, $EP_{i_2}^P$, $\cdots$, $EP_{i_x}^P$, $i_1 < i_2 < \cdots < i_x \leq i$, and these most general emerging patterns of D^N: $EP_{j_1}^N$, $EP_{j_2}^N$, $\cdots$, $EP_{j_y}^N$, $j_1 < j_2 < \cdots < j_y \leq j$. The next step is to calculate two scores for predicting the class label of T. Suppose we use k ($k \ll i$ and $k \ll j$) top-ranked most general emerging patterns of D^P and D^N. Then we define the score of T in the D^P class as

$$score(T, D^P) = \sum_{m=1}^{k} \frac{frequency(EP_{i_m}^P)}{frequency(EP_m^P)},$$

and the score in the D^N class is similarly defined in terms of $EP_{j_m}^N$ and EP_m^N. If $score(T, D^P) > score(T, D^N)$, then T is predicted as the class of D^P. Otherwise it is predicted as the class of D^N. We use the size of D^P and D^N to break tie. The PCL classifier has proved to be a good tool for analysing gene expression data and proteomic data [22, 41, 23]. Here we use it on recognizing translation initiation sites.

Example 7. We apply PCL, with $k = 10$, in the same 3-fold cross validation using the top 10 features selected in each fold as per Example 6. An accuracy of 87.7% is obtained as shown in the table below. This accuracy of 87.7% is comparable to the 89.4% in our earlier work [42] using a similar approach of feature generation and feature selection followed by the application of a machine learning method, except that our earlier work considered NB, SVM, and C4.5 on the same data set and used only nucleotide k-mers. It also compares well with the 78% sensitivity on TIS, 87% sensitivity on non-TIS, and 85% accuracy reported by Pedersen and Nielsen [30] on the same data set. It also compares favourably with the 69.9% sensitivity on TIS, 94.1% sensitivity on non-TIS, and 88.1% accuracy reported by Zien et al [43] on the same data set. Further improvement can be obtained by incorporating a form of distance or scanning rule [17, 42].

	1st fold	2nd fold	3rd fold	overall
no. of TIS (a)	1104	1104	1104	3312
correct predictions (b)	926	936	944	2806
wrong predictions (c)	178	168	160	506
sensitivity (a/b)	83.9%	84.8%	85.5%	84.7%
no. of non-TIS (d)	3355	3354	3354	10063
correct predictions (e)	2998	2928	2996	8922
wrong predictions (f)	357	426	358	1141
sensitivity (e/d)	89.4%	87.3%	89.3%	88.7%
accuracy ($[b+e]/[a+d]$)	88.0%	86.7%	88.4%	87.7%

□

Beyond good accuracy, PCL also has the advantage of producing comprehensible decision rules derived from the top emerging patterns. E.g., the emerging pattern $\{\texttt{UP}-\texttt{ATG} = Y, \texttt{DOWN}-\texttt{STOP} = Y, \texttt{DOWN}-\texttt{A} = 0\}$ has 10% support in our non-TIS sequence segments and 0% support in our TIS sequence segments. Then a rule can be derived from this emerging pattern as: *If the sequence segment has an in-frame ATG up-stream from the candidate TIS, and an in-frame stop codon down-stream from the candidate TIS, and there is no occurrence of any codons for the amino acid A, then this candidate TIS is a non-TIS with 10% probability and is a TIS with 0% probability.*

3.6 A Practical Challenge

Given two datasets D^P and D^N, an emerging pattern—to be used by PCL—is a pattern such that its frequency in D^P (or D^N) is non-zero but in the other dataset is zero (i.e., infinite ratio between its support in D^P and D^N), and none of its proper subpattern is an emerging pattern (i.e., most general). If D^P and D^N have n Boolean attributes, then there are 2^n possible patterns. Hence a naive method to extract emerging patterns requires 2^n scans of the dataset. A more efficient method for extracting emerging patterns is therefore crucial to the operation of PCL. Let us begin with a theoretical observation:

Theorem 2 (Cf. [12]). *The emerging patterns from D^P to D^N—all of them together, not just most general the ones—form a convex space.* $\square$

It is known that a convex space C can be represented by a pair of borders $\langle L, R \rangle$, so that (a) L is anti-chain, (b) R is anti-chain, (c) each $X \in L$ is more general than some $Z \in R$, (d) each $Z \in R$ is more specific than some $X \in L$, and (e) $C = \{Y \mid \exists X \in L, \exists Z \in R, X \subseteq Y \subseteq Z\}$. Actually, L are the most general patterns in C, and R the most specific patterns in C. We can write $[L, R]$ for C. Observe that

Proposition 1. *Suppose D^P and D^N have no duplicate and no intersection and are of the same dimension. Then the set of emerging patterns from D^P to D^N is given by* $[\{\{\}\}, D^P] - [\{\{\}\}, D^N]$. $\square$

Let $D^P = \{A_1, \ldots, A_n\}$ and $D^N = \{B_1, \ldots, B_m\}$. Then $[\{\{\}\}, D^P] - [\{\{\}\}, D^N] = [\{\{\}\}, \{A_1, \ldots, A_n\}] - [\{\{\}\}, \{B_1, \ldots, B_m\}] = [L, \{A_1, \ldots, A_n\}]$, where $L = min(\bigcup_i^n \{\{ s_1, \ldots, s_m\} \mid s_j \in A_i - B_j, 1 \leq j \leq m\})$. This definition of L is something we know how to optimize [11, 12] using datamining ideas from Max-Miner's look-ahead [3], Apriori's candidate generation [1], and level-wise search [27].

4 Another Important Role of Databases: Laboratory Workflow

We have seen the role that database research has played in data integration and knowledge discovery in modern molecular biology research. We have also described two technologies that were very original when they were first developed. The first is Kleisli, introduced in 1994. It is the first broad-scale data integration system that employs the nested relational data model, an explicit data exchange format, and a mediator-wrapper architecture. These features greatly facilitated the incorporation of numerous biological data sources and applications into Kleisli. The second is PCL—and the idea of emerging patterns, introduced in 1998—which is a machine learning method quite distinct from traditional machine learning methods. It produces highly human-understandable rules and also achieves very good accuracy.

To wrap up this paper, we discuss a third area where database research has a key role to play—the workflow of laboratory information management systems (LIMS). LIMS are deployed in genomic laboratories right where the raw genomic data are generated. The informatics challenge there is to control laboratory workflow, while keeping track of the mountains of data being generated in a consistent and accountable fashion. A

LIMS is thus a specialized database management system that focuses on the management of the two key aspects of a genome laboratory: Data and process.

Data have been a key achievement of the genome era—an era that has resulted in a multitude of new technologies that enable researchers to, say, sequence entire genomes [20] or scan tens of thousands of genes in a single chip experiment [15]. Such technological advancements have brought about challenging data management requirements for LIMS: (a) Voluminous data management. LIMS databases should be highly scalable as genome technologies have been designed to generate data *en masse*, and typical data entries in genome laboratories are often large capacity data items such as images and genetic sequences. (b) Heterogeneous data format handling. Laboratory data are often captured in the LIMS in a processed form (e.g., normalized peak heights instead of the entire electropherograms), requiring that the LIMS be able to access the proprietary file formats of different laboratory equipment to perform the necessary data extraction. A good LIMS should also capture useful data-associated user annotations, usually descriptive in nature and is best represented as free text. (c) Persistent and secure data archiving. The computerization of the laboratory and the emergence of bioinformatics have resulted in more and more laboratory data residing on computer systems in various digital formats. A file repository component must be included with a LIMS to persist, secure, and organize data files captured in the laboratory procedures. The recent FDA's requirement on "21 CFR Part 11" compliance has made it urgently necessary for drug companies to implement persistent and secure electronic archiving of laboratory data. A LIMS must therefore have built-in mechanisms to ensure the authenticity and integrity of the data.

Management of laboratory processes is the other focus of LIMS: (a) Sample tracking. Most laboratories implement a barcode system for tracking samples physically (e.g., blood, tissue, DNA). "Logical" tracking of laboratory samples, on the other hand, can be a complicated data management task in genome laboratories: A sample may be duplicated into multiple copies, extracted into various smaller fragments, loaded with others onto a single plate or array, reloaded with different groups of samples for another experimental procedures, and so on. LIMS databases must model such complex transformations of genomic objects. (b) Protocol tracking. Experimental protocols should be recorded in the LIMS and associated with the data generated, so that what has been done to a laboratory sample can be explained or redone. However, in an emerging field such as genomics, experimental protocols are continuously being re-engineered. LIMS databases must support an efficient versioning scheme for defining and updating protocols and tracking the changes made. (c) Inventory tracking. To maintain high throughput in laboratories, inventory tracking is an essential LIMS functionality so that the necessary laboratory materials (e.g., reagents, test-tubes, plates) are in stock and valid (for those with expiry dates). This means that all laboratory procedures must be modeled in fine details in the LIMS so that the database can automatically estimate current inventory based on usage.

Other LIMS-related operational database requirements include: (a) Instrument integration and interfacing. Integration is a key issue for LIMS, as it is not uncommon to employ multiple systems to conduct experiments in genome laboratories, and the LIMS is often expected to automate the operation of some of the laboratory processes in addition to mere data extraction from the different machines. (b) User and project management. A LIMS is often an enterprise solution in a genome institution, catering to a mixed community of users from laboratory technicians to research scientists and managers, each having his own use requirements in different experimental projects. The LIMS must therefore satisfy the different user needs, and implement appropriate access control schemes for the different categories of users and projects.

Database research has enabled the transition of traditional biology into data-intensive genomics, by meeting the many challenging requirements in data generation, capture, and annotation through technological development such as Kleisli and operationally scalable implementation of LIMS. With the coming-of-age of bioinformatics in the genome era, computer analysis has become an integral part of genome knowledge discovery [34]. Post-genome LIMS not only impacts people within the wet laboratory but also extends into many other areas of the new biological discovery process, including bioinformaticists and biostatisticians in the "dry" labs. Traditional LIMS has focused on tracking data and processes in the physical environment of the wet laboratories; the post-genome database adventures in LIMS would be to also track the data and processes in the virtual environment of the dry labs, and relating them with those in the wet labs in one single consistent database system.

Acknowledgements. We thank Huiqing Liu for help in the TIS recognition.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB'94*, pp 487–499.
2. P. G. Baker and A. Brass. Recent development in biological sequence databases. *Curr. Op. Biotech.*, 9:54–58, 1998.
3. R. J. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD'98*, pp 85–93.
4. P. Buneman et al. Comprehension syntax. *SIGMOD Record*, 23:87–96, 1994.
5. P. Buneman et al. Principles of programming with complex objects and collection types. *TCS*, 149:3–48, 1995.
6. J. Chen et al. The Kleisli query system as a backbone for bioinformatics data integration and analysis. In *Bioinformatics: Managing Scientific Data*, Morgan Kaufmann. To appear.
7. T.M. Cover and P.E. Hart. Nearest neighbour pattern classification. *IEEE Trans. Info. Theory*, 13:21–27, 1967.
8. L. Damas and R. Milner. Principal type-schemes for functional programs. In *POPL'82*, pp 207–212.
9. S. Davidson et al. BioKleisli: A digital library for biomedical researchers. *Intl. J. Digit. Lib.*, 1:36–53, 1997.
10. Department of Energy. *DOE Informatics Summit Meeting Report*, 1993.
11. G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *KDD'99*, pp 15–18.
12. J. Li et al. The space of jumping emerging patterns and its incremental maintenance algorithms In *ICML'00*, pp 551–558.
13. U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI'93*, pp 1022–1029,
14. D. Gerhold et al. DNA chips: promising toys have become powerful tools. *Trends Biochem. Sci.*, 24:168–173, 1999.

15. T.R. Golub et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.
16. L.M. Haas et al. DiscoveryLink: A system for integrated access to life sciences data sources. *IBM Systems Journal*, 40:489–511, 2001.
17. A.G. Hatzigeorgiou. Translation initiation start prediction in human cDNAs with high accuracy. *Bioinformatics*, 18:343–350, 2002.
18. G. Jaeschke and H. J. Schek. Remarks on the algebra of non-first-normal-form relations. In *PODS'82*, pp 124–138.
19. M. Kozak. An analysis of 5'-noncoding sequences from 699 vertebrate messenger RNAs. *NAR*, 15:8125–8148, 1987.
20. E.S. Lander et al. Initial sequencing and analysis of the human genome. *Nature*, 409:861–921, 2001.
21. P. Langley et al. An analysis of Bayesian classifier. In *AAAI'92*, pp 223–228.
22. J. Li et al. Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (ALL) patients. *Bioinformatics*, 2002. To appear.
23. J. Li and L. Wong. Geography of differences between two classes of data. In *PKDD'02*, pp 325–337.
24. L. Libkin and L. Wong. Query languages for bags and aggregate functions. *JCSS*, 55(2):241–272, October 1997.
25. H. Liu and R. Sentiono. Chi2: Feature selection and discretization of numeric attributes. In *Proc. IEEE 7th Intl. Conf. on Tools with Artificial Intelligence*, pp 338–391, 1995.
26. A. Makinouchi. A consideration on normal form of not necessarily normalised relation in the relational data model. In *VLDB'77*, pp 447–453.
27. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1:241–258, 1997.
28. Y. Papakonstantinou et al. Object exchange across heterogenous information sources. In *ICDE'95*, pp 251–260.
29. P. Pearson et al. The GDB human genome data base anno 1992. *NAR*, 20:2201–2206, 1992.
30. A.G. Pedersen and H. Nielsen. Neural network prediction of translation initiation sites in eukaryotes: Perspectives for EST and genome analysis. *ISMB*, 5:226–233, 1997.
31. J.R. Quinlan. *C4.5: Program for Machine Learning*. Morgan Kaufmann, 1993.
32. D. E. Rumelhart et al. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
33. G. D. Schuler et al. Entrez: Molecular biology database and retrieval system. *Methods Enzymol.*, 266:141–162, 1996.
34. D.B. Searls. Using bioinformatics in gene and drug discovery. *DDT*, 5:135–143, 2000.
35. S.J. Thomas and P.C. Fischer. Nested relational structures. In *Advances in Computing Research: The Theory of Databases*, pp 269–307, 1986.
36. V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
37. P. Wadler. Comprehending monads. *Math. Struct. Comp. Sci.*, 2:461–493, 1992.
38. L. Wong. Normal forms and conservative extension properties for query languages over collection types. *JCSS*, 52:495–505, 1996.
39. L. Wong. Kleisli, a functional query system. *JFP*, 10:19–56, 2000.
40. L. Wong. Kleisli, its exchange format, supporting tools, and an application in protein interaction extraction. In *BIBE'00*, pp 21–28.
41. E.J. Yeoh et al. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, 1:133–143, 2002.
42. F. Zeng et al. Using feature generation and feature selection for accurate prediction of translation initiation sites. In *GIW'02*. To appear.
43. A. Zien et al. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16:799–807, 2000.