Bioinformatics Integration Simplified: The Kleisli Way

Limsoon Wong
Kent Ridge Digital Labs
21 Heng Mui Keng Terrace, Singapore 119613
Email: limsoon@krdl.org.sg

7 December 1998

Abstract

The recent explosion of genomic information has been fueled by engineering and technological advances. However, as the amount of information grows the challenge becomes one of managing and making sense of the data. In fact, biological discovery is increasingly performed in a "dry and wet" paradigm, in which hypotheses are first formulated by analyzing existing data sources and subsequently confirmed by experiments. This paradigm relies on the development of a number of tools that require advances in the fundamentals of computer science as well as careful engineering. "Dry" experiments inevitably involve integrating and transforming data souces so that they can be analyzed or "mined" with the appropriate software. Integration of these data sources is complicated by the fact that they use different formats and require different languages for accessing them. KRIS, better known as Kleisli, is a bioinformatics integration system designed to scale this "tower of babel." The power of Kleisli is reviewed here through a series of examples.

Keywords: Kleisli, data integration, bioinformatics, protein patent.

Introduction

Many problems in modern bioinformatics involve (a) accessing complex heterogeneous data sources that are geographically dispersed, (b) multiple sequential steps, and (c) passing information smoothly between these steps. Simple retrieval of data is not sufficient for modern bioinformatics. With the rapid growth of experimental data, in order to investigate a specific biological problem, the challenge is how to automate the process of manipulating and re-structuring of the information derived from various databases. This may require combining data derived from multiple public sources and local (private) sources and feeding the retrieved data into various application programs such as gene finding, protein structural prediction, functional domain or motif identification, phylogenetic tree construction, etc. All these procedures require specific input data sets and data formats. As observed by Baker and Brass [1], many existing biology data retrieval systems[2, 3, etc.] are not fully up to the demand of flexible and painless data integration. This is where the power of Kleisli comes into play.

Kleisli[4] is a powerful data integration system that interfaces to a large number of data sources relevant to bioinformatics and uses a self-describing data model to allow data derived from different sources to be flexibly combined. A partial list of these data sources can be found at

`http://adenine.krdl.org.sg:8080/publications/drivers.html`

There are more than two hundred biological databases and servers in the Internet[5, 6, etc.] Merely providing an interface to a collection of databases and analysis software is often not useful if it requires tedious programming to make use of the interface, as is the case with CORBA [7]. Kleisli goes one step further and provides a high-level query language called Collection Programming Language (CPL), based on elegant mathematical principles[8]. CPL offers a rich data model and many high-level operators to express complex queries and tranformations on these biology databases and analysis software in an extremely straightforward manner that does not require extensive programming skill.

Many complicated bioinformatics queries involving multiple databases and analysis software in multiple steps have a simple expression in CPL and are efficiently executed by Kleisli. In order to properly appreciate the virtue of the system, it is necessary to see some real-life examples. I describe below two example queries. The first example asks what proportion of human mature peptides have prolines at their N-terminal. This simple example serves as a quick introduction to the basic syntax of CPL. The second example asks what other protein sequences in the same superfamily of a given protein sequence have been patented. These examples exercise many aspects of Kleisli and involves integration across Entrez[2], SCOP[9], WU-BLAST2[10], patents, proteins, feature tables, etc. I hope the succintness of these examples is sufficient illustration of the power, flexibility, and simplicity of the Kleisli system.

Example: Proline at N-Terminal

The first of our two examples is this query: *What proportion of mature peptides from human have prolines at their N-terminal?* Its implementation in Kleisli/CPL is given below.

```
1.   {string-span (x.#sequence, f.#start, f.#end)
2.   | \x <- aa-get-seqfeat-general "mature AND peptide AND human[Organism]",
3.     \f <- x.#feature, f.#name string-islike "%mat_peptide%" }
4. before (\mat-peptides =>
5.   {x | \x <- mat-peptides, x string-islike "P%" }
6. before (\prolines =>
7.   set-roughcount (prolines) / set-roughcount (mat-peptides)));
```

The syntax and meaning of CPL program is straightforward. e `before` $(\backslash x \Rightarrow\ e')$ means evaluate e and set the variable x to the result and then evaluate e' (which may use this value of x.) $\{e \mid \backslash x \leftarrow\ S,\ C\}$ means the set of e such that x is in the set S and the condition C is true; or in more prosaic words, iterate over each item x in the set S, if the condition C is true for this x, then evaluate e (which may depends on the value of this x) and include the result in the output set. $e.\#l$ means the value at the l field of the record e. Operators such as `aa-get-seqfeat-general` are provided in the Kleisli system to interface to various bioinformatics sources such as Entrez. For more detail, consult the CPL Reference Manual[11].

We now return to explaining the implementation of the proline query above. It basically consists of three steps.

1. Entrez is accessed to obtain GenPept reports of human amino acid sequences (line 2). For each report `x` of a sequence, its feature table is inspected to extract feature entries (line 3). For each feature entry `f`, we check if it annotates a mature peptide (line 3). If it is a mature peptide, the start and end positions are extracted from the annotation and are used to extract that segment of the sequence of `x` (line 1). The resulting sequence segments are assigned to the set `mat-peptides` (line 4).

2. For each sequence segment `x` in the set `mat-peptides`, we check if it starts with the letter `P` (line 5). If so, `x` is a mature peptide that has a proline at its N-terminal. These resulting sequence segments are assigned to the set `prolines` (line 6).

3. We just count the number of items in the set `prolines` and the number of items in the set `mat-peptides`. Their ratio gives the desired answer (line 7).

Example: Patented Proteins in Superfamily

Assessing the patent potential of a sequence is of practical importance. The second of our two examples is an interesting query on this subject: *Suppose we are given a protein sequence* `SEQ`*. List patented sequences that belong to the same superfamily as* `SEQ`. By a superfamily, we mean sequences that have the same tertiary structure and similar functions, but they do not have to have similar sequences. The implementation in Kleisli/CPL is given below. A full exposition to the complement of this question—list unpatented sequences in the same superfamily as `SEQ`—was given in [12].

```
 1. localblast-blastp (#name: "scop-blast", #db: "scopseq");
 2. localblast-blastp (#name: "pat-blast", #db: "patseq");
 3. setindex-access (#name: "sid2seq", #file: "scopseq", #key: "#sid");
 4. scop-add "scop";
 5. {(#sf: (#desc: xinfo.#desc.#sf, #hit: x.#accession, #pscore: x.#pscore),
 6.   #bridge: (#hit: s, #patent: p.#title, #pscore: p.#pscore))
 7. | \x <- process SEQ using scop-blast, x.#pscore <= PSCORE,
 8.   \xinfo <- process <#sidinfo: x.#accession> using scop,
 9.   \s <- process <#numsid: xinfo.#type.#sf> using scop,
10.   \y <- process <#key: s> using sid2seq,
11.   \p <- process y.#seq using pat-blast, p.#pscore <= PSCORE };
```

The implementation consists of two steps.

1. We need to use SCOP[9] to obtain superfamily classification. We also need access to patented sequences (we got them in the file `patseq`) and to representative sequences for the various superfamilies in SCOP (we got them in the file `scopseq`). So we establish a WU-BLAST2[10] connection `scop-blast` to `scopseq` (line 1), a WU-BLAST2 connection `pat-blast` to `patseq` (line 2), and a connection `scop` to SCOP (line 4). To provide random access to representative sequences of SCOP superfamilies given their SCOP identifier (`sid`), we also establish the database index `sid2seq` (line 3). Having set all these connections up, we are ready to do the real work.

2. We BLAST the given sequence `SEQ` against SCOP superfamily representative sequences to look for a hit `x` at a strong threshold `PSCORE` (line 7). `SEQ` is regarded to be in the same superfamily as `x`. We look up `x`'s record from SCOP (line 8). From this record, we ask SCOP for all the identifiers of representatives in the superfamily of `x` (line 9). Each representative `s` serves as a "bridge" to help us identify the desired patented sequences. For each such representative `s`, we use our index to look up its sequence `y` (line 10). We BLAST this sequence `y` against the patented sequences; those hits at the strong threshold `PSCORE` are the patented sequences we want (line 11). The relevant information (the superfamily information of `x`, the bridge `s`, and the patent information `p`) are returned in the output set (lines 5–6).

Discussion

Let me end this paper by making a brief comparison of Kleisli to two previous systems. It is fair to say that SRS [3] is one of the more successful tools for integrating biology databases. It is essentially a link-based interface. It is very convenient to use for simple operations. However, it offers no facility for flexible transformation of data from these sources, an important aspect in more advanced bioinformatics applications. As a consequence, one needs a significant amount of manual work if one relies mostly on SRS for data integration. TAMBIS [13] is slightly more comparable to Kleisli. It is an interface to many sources that come with a rather fixed set of queries that offer more slightly interesting integration of the sources than SRS. The kind of queries that can be expressed basically have to fall into templates anticipated by TAMBIS' designers. Thus, the expressible queries are much more limited than in Kleisli.

In compensation to the limitation on its query expressive power, its queries can be formulated using a looser syntax and is supported by a knowledge-driven user interface. This could be an advantage for less skillful programmers. However, it is worth pointing out that TAMBIS actually runs on top of an old version of Kleisli, demonstrating the value of Kleisli as a substrate for constructing other integration systems.

The Kleisli system is now made commercially available by KrisTech Inc. in California under the name KRIS. KRIS stands for both "Kleisli-Related Integration System" and "Kent Ridge Integration System" to distinguish it from the early prototype developed several years ago at the University of Pennsylvania. This new name KRIS is chosen also because it means the "wavy-bladed" type of daggers original to the Malay archipelago where the industrial-strength version of Kleisli was developed—Singapore.

References

[1] Baker PG, Brass A. Recent development in biological sequence databases. Current Opinion in Biotechnology 1998; 9:54–58.

[2] Schuler GD, Epstein JA, Ohkawa H, Kans JA. Entrez: Molecular biology database and retrieval system. Methods in Enzymology 1996; 266:141–162.

[3] Etzold T, Argos P. SRS: Information retrieval system for molecular biology data banks. Methods in Enzymology 1996; 266:114–128.

[4] Davidson S, Overton C, Tannen V, Wong L. BioKleisli: A digital library for biomedical researchers. International Journal of Digital Libraries 1997; 1:36–53.

[5] Ashburner M, Goodman N. Informatics—genome and genetic databases. Current Opinion in Genetics & Development 1997; 7:750–756.

[6] Karp PD. Metabolic databases. TIBS 1998; 23:114–116.

[7] Siegel J. CORBA: Fundamentals and Programming. Wiley, New York, 1997.

[8] Buneman P, Naqvi S, Tannen V, Wong L. Principles of programming with complex objects and collection types. Theoretical Computer Science 1995; 149:3–48.

[9] Murzin A, Brenner SE, Hubbard T, Chothia C. SCOP: A structural classification of protein database for the investigation of sequences and structures. Journal of Molecular Biology 1995; 247:536–540.

[10] Altschul SF, Gish W. Local alignment statistics. Methods in Enzymology 1996; 266:460–480.

[11] Wong L. The Collection Programming Language Reference Manual, 1998. Kent Ridge Digital Labs, 21 Heng Mui Keng Terrace, Singapore 119613.

[12] Chen J, Zhang L, Wong L. A protein patent query system powered by Kleisli. Proceedings of ACM SIGMOD International Conference on Management of Data 1998; 593–595.

[13] Baker PG, Brass A, Bechhofer S, Goble C, Paton N, Stevens R. TAMBIS—transparent access to multiple bioinformatics information sources. Intelligent Systems for Molecular Biology 1998, 6:25–34.